REPORT RESUMES

ED 020 660 EM 000 242

THE COMPUTER AS AN AID IN TEACHING MATHEMATICS, AN INSTRUCTIONAL BULLETIN, GRADES 7-10.
BY- LERNER, SEYMOUR
LOS ANGELES CITY SCHOOLS, CALIF.

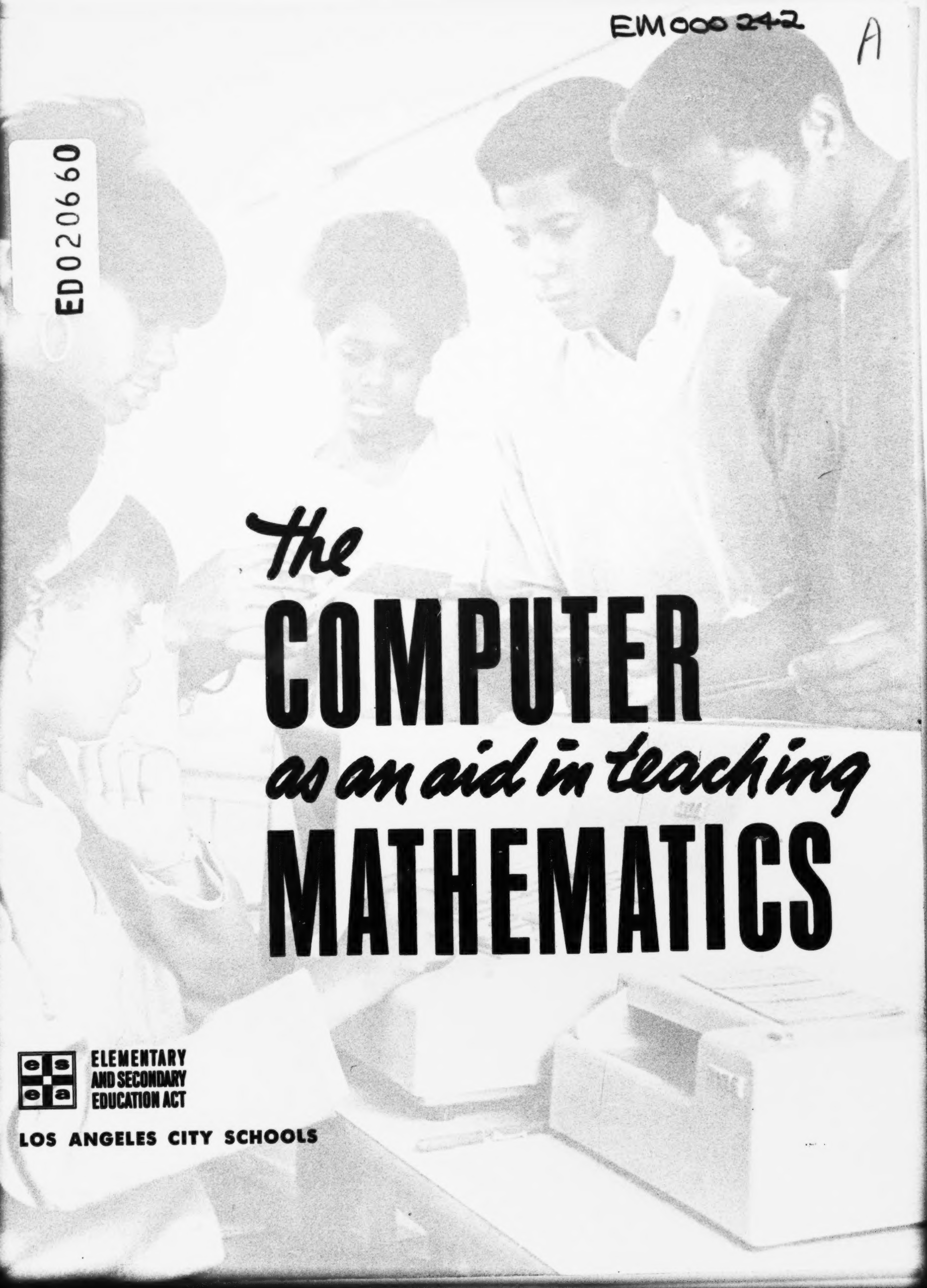
REPORT NUMBER ESEA-5 PUB DATE 67
EDRS PRICE MF-$0.50 HC-$4.36 107P.

DESCRIPTORS- EDUCATIONAL OBJECTIVES, SELF ESTEEM, STUDENT BEHAVIOR, CONCEPT FORMATION, MATHEMATICAL CONCEPTS, *COMPUTERS, *COMPUTER ASSISTED INSTRUCTION, *MATHEMATICS INSTRUCTION, DIGITAL COMPUTERS, ANALOG COMPUTERS, CONCEPT TEACHING, PROBLEM SOLVING, INDIVIDUAL STUDY, STUDY HABITS, TRANSPARENCIES, WYLE SCIENTIFIC, FORTRAN

THIS GUIDE EXPLAINS HOW A COMPUTER MIGHT BE USED IN MATHEMATICS EDUCATION IN CONJUNCTION WITH THE REGULAR COURSE OF STUDY. STARTING WITH THE PREMISE THAT A CHILD WHO LEARNS TO OPERATE A COMPUTER IS LED BY HIS NEW SELF ESTEEM TOWARD A GREATER INTEREST IN HIS COURSES, THE AUTHORS STATE THAT COMPUTER EDUCATION WILL ALSO MEET THESE OBJECTIVES--(1) INCREASE SKILLS, (2) AID IN CONCEPT UNDERSTANDING, (3) INCREASE PROBLEM SOLVING ABILITY, (4) STIMULATE INDIVIDUAL RESEARCH, AND (5) DEVELOP DESIRABLE HABITS. THE OPERATIONAL PROCEDURES OF COMPUTER USE AND THEIR UNDERLYING IDEA OF FLOW CHARTING ARE THEN EXPLAINED IN THE GUIDE. THESE INCLUDE THE COMPUTER SYSTEM, CLARIFICATION OF COMPUTER CAPABILITIES, CHARACTERISTICS OF THE WYLE SCIENTIFIC, AND LOOPING PROCESSES. PRACTICE PROBLEMS ARE PROVIDED AT THE END OF EACH TOPICAL DISCUSSION, AND A SEPARATE CHAPTER PROVIDES EXAMPLES AND METHODS OF ATTAINING COMPUTER EDUCATION OBJECTIVES. ALSO EXPLAINED ARE DIFFERENT NUMBER NOTATION SYSTEMS, AIDS TO USING COMPUTERS, AND FORTRAN PROGRAMING. APPENDICES INCLUDE SOME POPULAR COMPUTER LANGUAGES, A GLOSSARY, A TABLE OF POWERS OF 2, AND A BIBLIOGRAPHY. TRANSPARENCY MASTERS ARE PROVIDED SEPARATELY. (JO)

EM000242 A

ED020660



the COMPUTER as an aid in teaching MATHEMATICS

ELEMENTARY
AND SECONDARY
EDUCATION ACT

LOS ANGELES CITY SCHOOLS

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

THE COMPUTER AS AN AID IN TEACHING MATHEMATICS

An Instructional Bulletin
Grades 7-10

Los Angeles City Schools
Division of Secondary Education
Specially-Funded Programs
Publication No. ESEA-5
1967

EM000 242

This publication was developed with funds provided by the federal government under Title I, Elementary and Secondary Education Act of 1965.

APPROVED:

EVERETT CHAFFEE
Associate Superintendent
Division of Instructional Planning and Services

ROBERT E. KELLY
Associate Superintendent
Division of Secondary Education

FOREWORD

In the very near future, it is probable that small computers such as the Wyle Scientific will be used in most secondary schools as an aid in teaching mathematics. All computer courses given today in our colleges and universities emphasize programing and data processing. A guide is needed which explains how a computer can be used in mathematics education in conjunction with the regular course of study. This publication has been designed to serve that need.

The objectives of using a computer in mathematics education and the methods of achieving those objectives with the use of a computer are detailed in this material. Also, a variety of problems that can be used in this type of program are included.

ROBERT E. KELLY
Associate Superintendent
Division of Secondary Education

ACKNOWLEDGMENTS

The guidance and contributions of staff members to the publication are acknowledged.

Appreciation is expressed to the teacher advisory committee:

Bette Andrews	Adams Junior High School
Warren Buchner	Gomper Junior High School
Alyce Jackson	Drew Junior High School
Leon Mason	Muir Junior High School
Gene McCallum	Markham Junior High School
Audrey McDaniel	Gompers Junior High School
Kazuo Ogawa	Adams Junior High School
John Otterness	Muir Junior High School
Gloria Riddick	Markham Junior High School
Russell Walton	Adams Junior High School
Roy Roberson	Drew Junior High School

Valuable guidance and assistance were provided by:

Division of Secondary Education

Keith F. Smith	Supervisor, Mathematics
Walter J. Lansu	Assistant Administrative Coordinator, Specially-Funded Programs
Al Lalane	Editor, Pilot Projects Specially-Funded Programs

Division of Instructional Planning and Services

Dr. Marian C. Herrick	Supervisor, Mathematics, Instructional Planning Branch

Particular appreciation is extended to Richard Lubin for his assistance in the writing of the manuscript and for many suggestions which contributed to its effectiveness.

The manuscript was written by SEYMOUR LERNER, Mathematics Consultant, Division of Secondary Education, Specially-Funded Programs.

HARRIETTE F. WILLIAMS
Pilot Projects Coordinator
Specially-Funded Programs

THOMAS O. LAWSON
Administrative Coordinator
Specially-Funded Programs

CONTENTS

Page

FOREWORD iii

ACKNOWLEDGMENTS iv

A PARTIAL LIST OF SYMBOLS USED vii

INTRODUCTION viii

UNITS 1

Unit 1 FLOW CHARTS 3

- Part I. The Need for Computer Education 3
- Part II. Objectives of Computer Education 4
- Part III. Flow Charting 5
- Part IV. Uses of Flow Charts 5

Unit 2 THE COMPUTER AND OPERATIONAL PROCEDURES 13

- Part I. Introduction 13
- Part II. Computer System 13
- Part III. Clarification of Computer Capabilities 15
- Part IV. Characteristics of the Wyle Scientific 17

Unit 3 LOOP PROCESSES 29

- Part I. Looping-The Single-Loop Processes 29
- Part II. The Nested-Loop Process 37

Unit 4 DEVELOPING EXAMPLES AND METHODS TO OBTAIN "OBJECTIVES OF COMPUTER PROGRAM" 45

Unit 5 NUMBER NOTATION SYSTEMS 53

- Part I. Decimal Number System 53
- Part II. Binary and Octal Number Systems 53
- Part III. Hexadecimal (base 16) Number System 55
- Part IV. Comparing Number Systems 55

Page

Unit 6 AIDS TO USING A COMPUTER 61

Part I. Subroutines 61
Part II. Transparencies 65

Unit 7 FORTRAN PROGRAMING 69

Part I. The Compiler 69
Part II. Fixed and Floating Points 69
Part III. Variables 70
Part IV. FORTRAN Symbols 70
Part V. Statements 71

APPENDIX 75

SOME POPULAR COMPUTER LANGUAGES 77

GLOSSARY 78

TABLE OF POWERS OF 2 81

BIBLIOGRAPHY 82

A PARTIAL LIST OF SYMBOLS USED

$\sum_{i=1}^{n} (X_i)^p$	Summation of $(X_i)^p$ from $i = 1$ to $i = n$, $i \in$ Integers ex. $\sum_{i=1}^{4} (X_i)^2 = (X_1)^2 + (X_2)^2 + (X_3)^2 + (X_4)^2$
$\approx$	Approximately equal to
$\neq$	Not equal to
$<$	Less than
$>$	Greater than
$\not<$	Not less than
$\not>$	Not greater than
$\leq$	Less than or equal to
$\geq$	Greater than or equal to
$B \leftarrow B-A$	The value of B takes the value of B-A
$B \leftrightarrow A$	The values of A and B are interchanged
$[\![x]\!]$	The greatest integer n such that $n \leq x$
IFF	If and only if

Fortran Symbols

+	Add
-	Subtract
*	Multiply
/	Divide
**	Exponentiation

INTRODUCTION

Let's talk about our pupils. Many of them are able but completely uninterested in mathematics as well as in school. We do not have to search far to find reasons for this attitude. This youngster is wise in the ways of the street. He knows exactly how far to push an adult to gain status in the eyes of his peers. His needs are real and evident if we will but search them out. He needs success. He needs peer approval. He needs self-esteem. He knows that none of these are available to him, in the classroom, by playing the game our way. He tried; but having failed has arrived at a set of rules which work for him.

He attends six or more classes per day and very quickly learns the personality quirks and the strengths and weaknesses of each of his teachers. He uses these to achieve his aims. Any pupil capable of formulating his own game, and making us play it, is not unintelligent. He hasn't played our game because he is not successful at it. It isn't fun. He plays his own game because he has a formula which works to give him a form of success and status. He could never and would never tell us what he is doing, but we should be able to recognize the pattern. How do we proceed to change his attitude and behavior? We recognize his primary need--self-esteem. We must change our behavior and our school patterns in order that he may achieve a high degree of success in the classroom situation.

Computers, at the present time, are held in awe by most adults and certainly by all pupils in our secondary schools. There seems to be an aura of mystery surrounding this machine and there appears to be a belief, albeit false, that only a genius can operate one. It is this very aspect of the computer that can be used very effectively, in a mathematics classroom, to motivate pupils. When he is convinced that he can operate a computer; when he becomes familiar with programing a computer, then, indeed, we "have him". His self-image soars and all frustrations due to past failures in mathematics seem to evaporate. "If I can operate a computer, then surely I can succeed in mathematics," he feels. Our task is clear. The game must be changed. No longer will our primary task be to present subject matter. No longer need the pupil reject this game and substitute his own. Now we play together. He can find immediate satisfaction in successful operation of a computer; something even the "brain" will envy. His self-esteem soars. He has peer acceptance. He is successful. He is happy. We are achieving our goals. Real learning can now take place.

We have made our first step; we have him hooked. Now we must very carefully formulate our activities to utilize his interest. It is our intention to examine, step by step, specific activities which work.

Realistically, this is not the only program which will provide the stimulation and opportunity for success needed. It is hopefully only a first step and will have to be refined and developed as the reader applies and adapts it to his own needs. It is hoped that your discoveries can be incorporated into this program as you find new, different, and more effective variations on these ideas.

The author feels that particular attention paid to Units I, II, and IV will reward the reader with a set of new, effective tools for motivating students. The remaining units are provided to encourage experimentation with related material but are not as essential to the reader's immediate needs.

UNITS

UNIT 1

FLOW CHARTS

Part I. The Need for Computer Education

The following statement best summarizes the need and future of computer education at the secondary school level:

> Every high school and college graduate has from 30 to 40 years of working life ahead of him. During this time, the accumulation of knowledge will be doubling every ten years (or less). This will have a tremendous effect on the working and living environments of most of these people. They should learn as early as possible as much as they can about the information processing technology so that they will be enlightened instead of frightened by the changes which will take place.
>
> Every engineer and scientist will have to learn how to use the computer as a tool to solve problems or retrieve information. Every lawyer and doctor will have to understand the possibilities of information retrieval and preliminary case-screening and analysis. Every banker and insurance man will have to understand something about statistics as well as information transmission and retrieval. Every technician should know something about the use of machines in his field, from drafting to printing, from photography to nursing, from social work to laboratory analysis.
>
> Every teacher will have to know how the data processing technology affects his field of interest so that he can improve his understanding, teaching, and research in his chosen field.
>
> Every businessman and administrator will have to know how to use machines to get information and to help him make the best possible decisions for the successful running of his business or organization.
>
> Every clerk and workman should understand something about the machines with which he will be working and which will, in one sense or another, be used in planning his work.*

---

*Report of the Conference on Computer Oriented Mathematics and the Secondary School. Washington, D.C. National Council of Teachers of Mathematics. (See pp. 49-52, Publication Procedures) May 24-25, 1963.

Part II. Objectives of Computer Education

1. Increase Skills

   The computer will be used as a computational device. Pupils will perform calculations to check the validity of algorithms. Computational skills are expected to increase.

2. Aid in Concept Understanding

   Computer programing methods will be used to generate concept formation and understanding.

3. Increase Ability in Problem Solving

   Settings will be provided to use the computer to solve problems. Different approaches to the solution of a problem will be explored.

4. Stimulate Individual Exploration and Research

   Because the use of the computer is a uniquely personal experience, it is the most powerful tool available to help the pupil learn to see himself as a future creator and explorer.

5. Develop Desirable Habits

   Habits of neatness, organization, and facility in working with basic instructions will be developed through the use of flow charting and programing.

6. Enhance Self-Image

   The computer is a very powerful tool in motivating interest in mathematics. The pupil always experiences success and satisfaction. Additional motivation involves the glamor of working with a computer.

A discussion of these objectives, as well as problems, and methods to obtain them, appears in Unit 4.

Note:

While flow charting is the logical approach to a study of computer programing, we must not lose sight of the pupil's pressing needs. It is highly recommended that the pupil experience a "hands on" situation, with the computer, the first day.

Operational procedures of the Wyle Scientific are presented in Unit 2.

Part III. Flow Charting

Flow charting is a technique of listing steps to be performed in sequence. When solving a problem with a digital computer, it is necessary to work out in advance definite logical procedures, or algorithms, which the machine can follow without further need for supervision. This is usually called a "program," and an excellent way of presenting the steps in a program is by means of a flow chart.

Flow charts are an integral part of all documentation, and as such, should accompany all documentation. Adherence to standard techniques for the preparation of flowcharts greatly increases effectiveness of communication between the programer analyst and the many groups with whom he deals. In order to encourage this standardization in the use of symbols, IBM (International Business Machines Corp.) has made available a Flowcharting Template (X20-8020). This template includes all the symbols necessary for both system and program flowcharts. For greater simplicity, we will not make use of all these symbols. The only symbols we will make use of, however, are shown in Figure 1-2. These symbols are used to represent what is occurring at any point in the flow.

Part IV. Uses of Flow Charts

Because of the variety of uses of flow charting in the mathematics class, apart from its use with the computer, it would perhaps be beneficial to digress from the main purpose in this unit and discuss some of these uses.

In order to depict the logical sequence of a flow chart, any adage or aphorism may be used. However, all students must be completely familiar with what we use.

We might start with the simple proverb "a stitch in time saves nine." After a discussion of this proverb and making sure that all pupils understand it, put it on the board in the following manner:

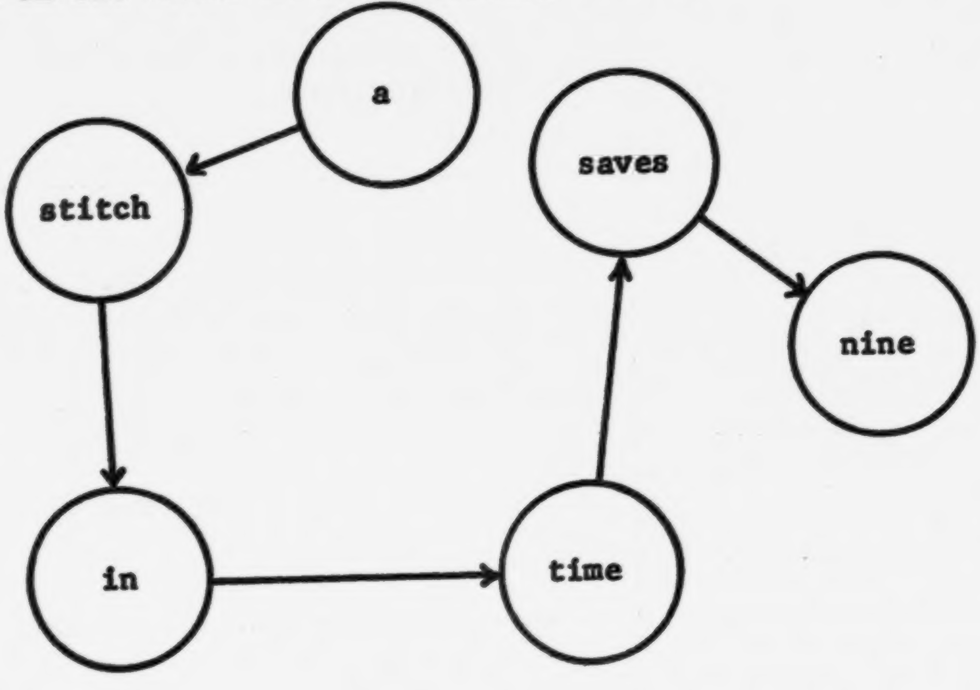


Figure 1-1

Note: Flow Chart Symbols are not used since it is not necessary to represent what is occurring in the flow.

Discuss with the pupils, the meaning of the arrows and the logical sequence of the flow chart. This could be a fascinating game and a lot of fun. Furthermore it lends itself very nicely to applications in many branches of mathematics. The following problems apply flow charting to increasing computational skills and to problem solving.

FLOW CHARTING SYMBOLS AND THEIR DESCRIPTIONS

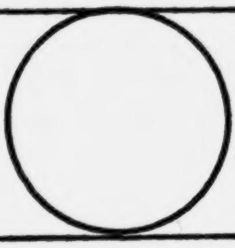
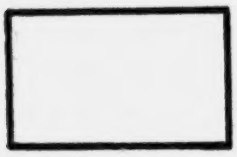
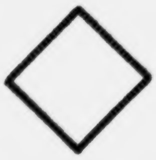
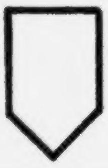
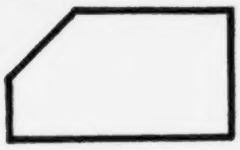
SYMBOL	MEANING
	Terminal - The beginning or end of a program
	Instruction - Any instruction which performs a processing function of the program
	Decision - Used when variable conditions offer alternate paths
← →↑↓	Flow Direction - The direction of flow of the diagram
	Offpage connector - Need to designate entry to or exit from a page
	Punched Card - Any type of punched card

Figure 1-2

Problem 1

Flow Chart the following problem in addition.

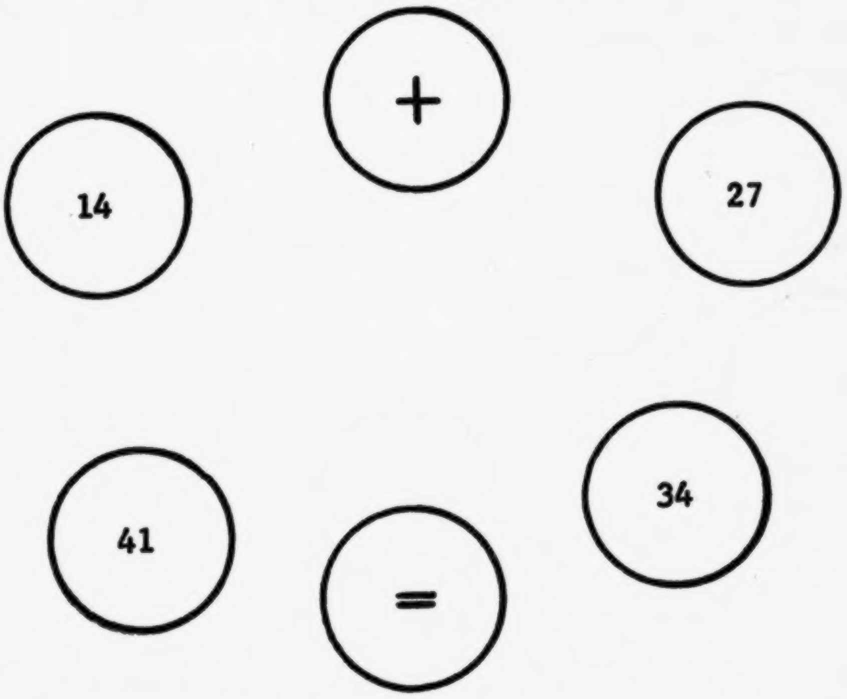


Figure 1-3

Many pupils will add the various combinations, in pairs of the numbers shown in Figure 1-3, This will give them needed drill in addition. All pupils should be led, eventually, to realize that either 34 or 41 must be the sum, and consequently we must add 14 and 27 to obtain the required answer. The solution to this problem follows:

Solution to Problem 1

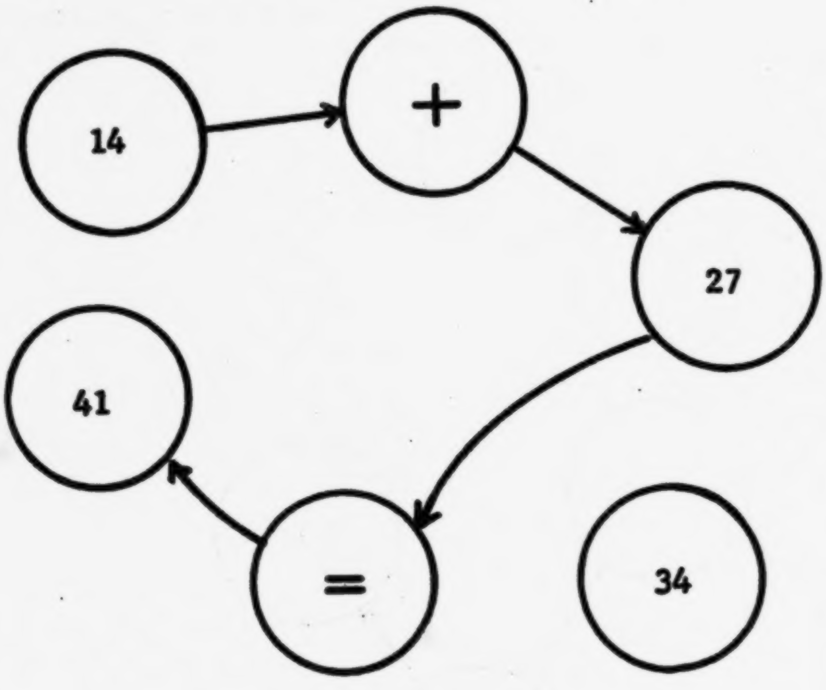


Figure 1-4

This type of flow chart can, of course, be applied to all operations of arithmetic. Vocabulary can be stressed by using the words equals, plus, sum, etc., instead of the symbols in the chart.

Problem 2

If you have \$1.60 and spend 50¢, how much of the \$1.60 will you have left? Solve this problem using the flow chart below and write the answer in the empty circle.

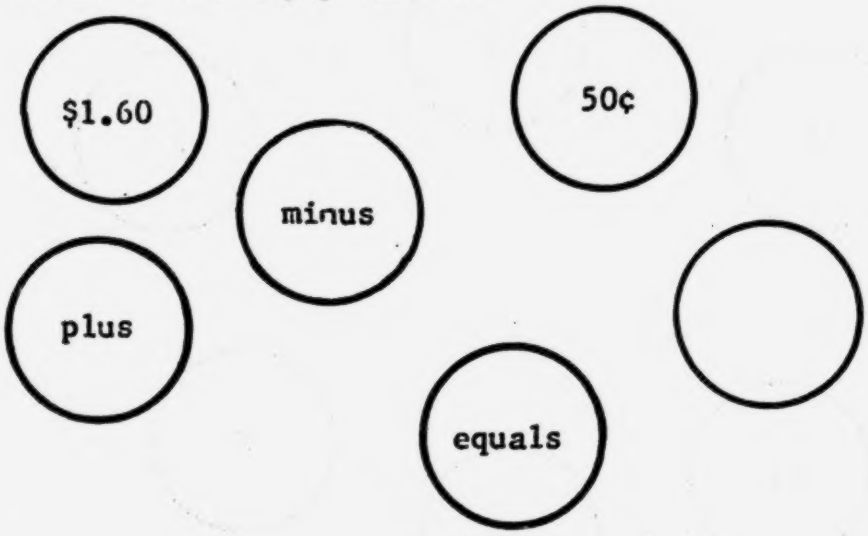


Figure 1-5

Figure 1-5 eliminates the difficulty of translating the word problem into a mathematical sentence. The only thing that the pupil must decide is whether to use the plus or the minus in this flow chart. Eventually the pupil should be able to solve this type of problem and set up his own flow chart. Problems can increase in complexity as the pupil gains more confidence in his ability to solve word problems.

Problem 3

If you have \$1.60 and spend one half of this amount, how much money will you have left?
This type of problem may be flow charted and solved through different approaches. Two of these approaches are shown in Figures 1-6 and 1-7.

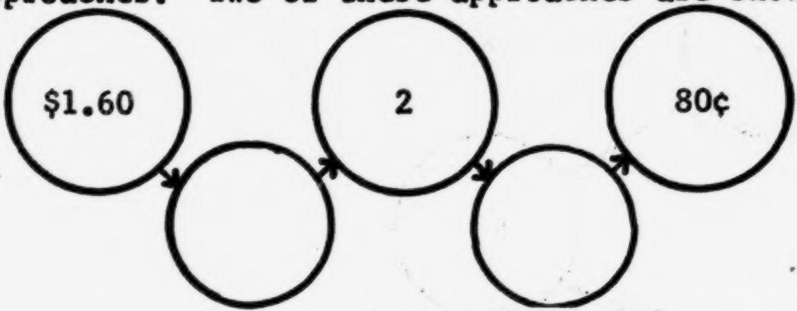


Figure 1-6

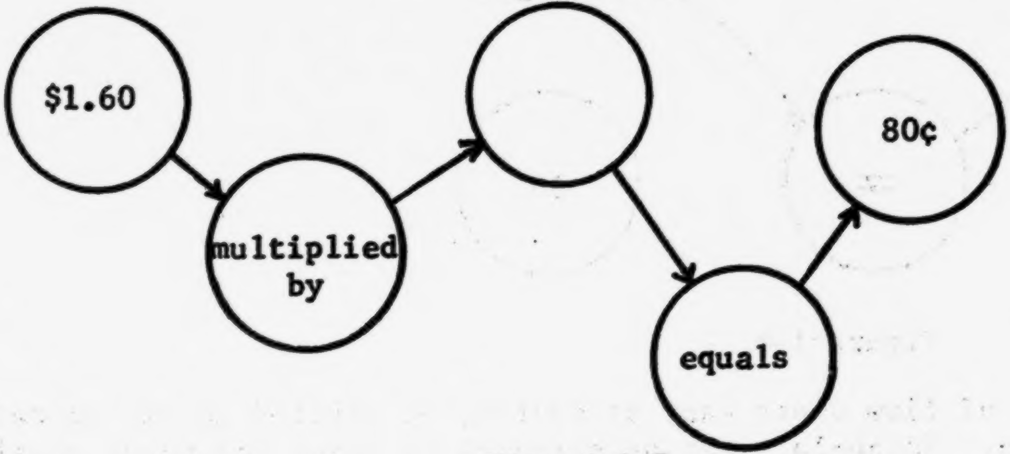


Figure 1-7

When flow charting a program to be used in a computer, it is necessary to represent what is occurring in the flow as well as giving the direction of the flow. For this representation the symbols in Figure 1-2 are used.

The following things are fun to flow chart and furnish an insight into programing.

1. Getting to school in the morning
2. Repairing a part of an automobile
3. Getting to work
4. Crossing a street
5. Eating dinner

Number 1 (Getting to school in the morning) could be flow charted in the following simple fashion.

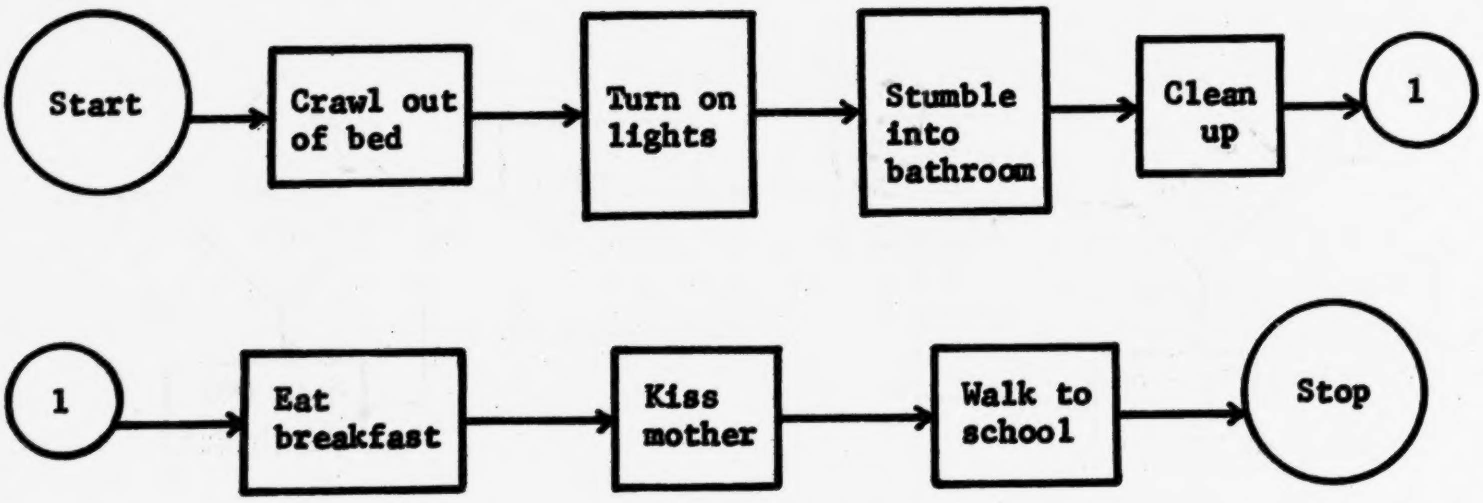


Figure 1-8

Figure 1-8 could get much more complicated by involving alarm clock, heat, decisions, etc.

The flow chart in Figure 1-9 displays the ramifications of decision making.

GETTING TO SCHOOL IN THE MORNING

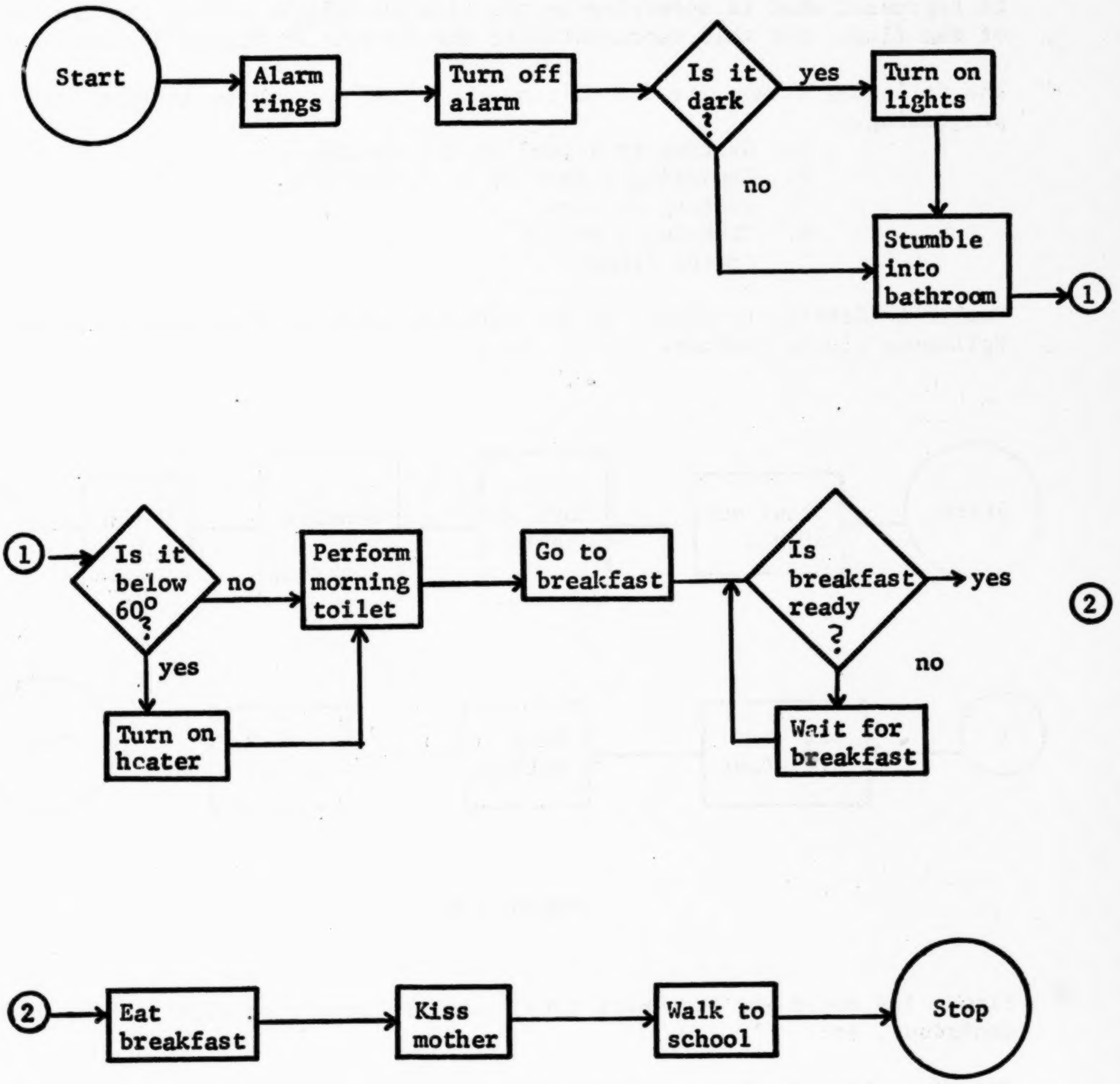


Figure 1-9

FLOW CHART FOR $y = 5X + 6$

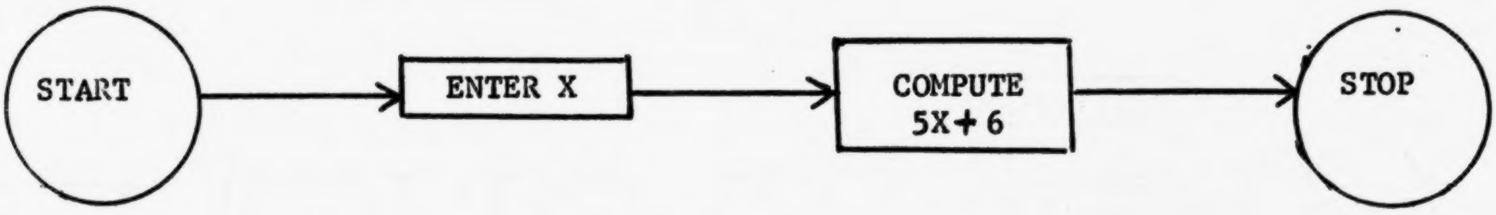


Figure 1-10

This instructional bulletin was written primarily to be used with the Wyle Scientific. However, all computers will require that each mathematics problem be broken down into discrete steps. The computer requires this because it cannot "think." We are ready to utilize this fact because it forces our pupil to examine his problem in order to find ways to communicate it to the computer. Even more, it forces him to examine the problem to think through, step by step, what has to be done to arrive at a solution.

EXPANDED FLOW CHART FOR $y = 5X + 6$

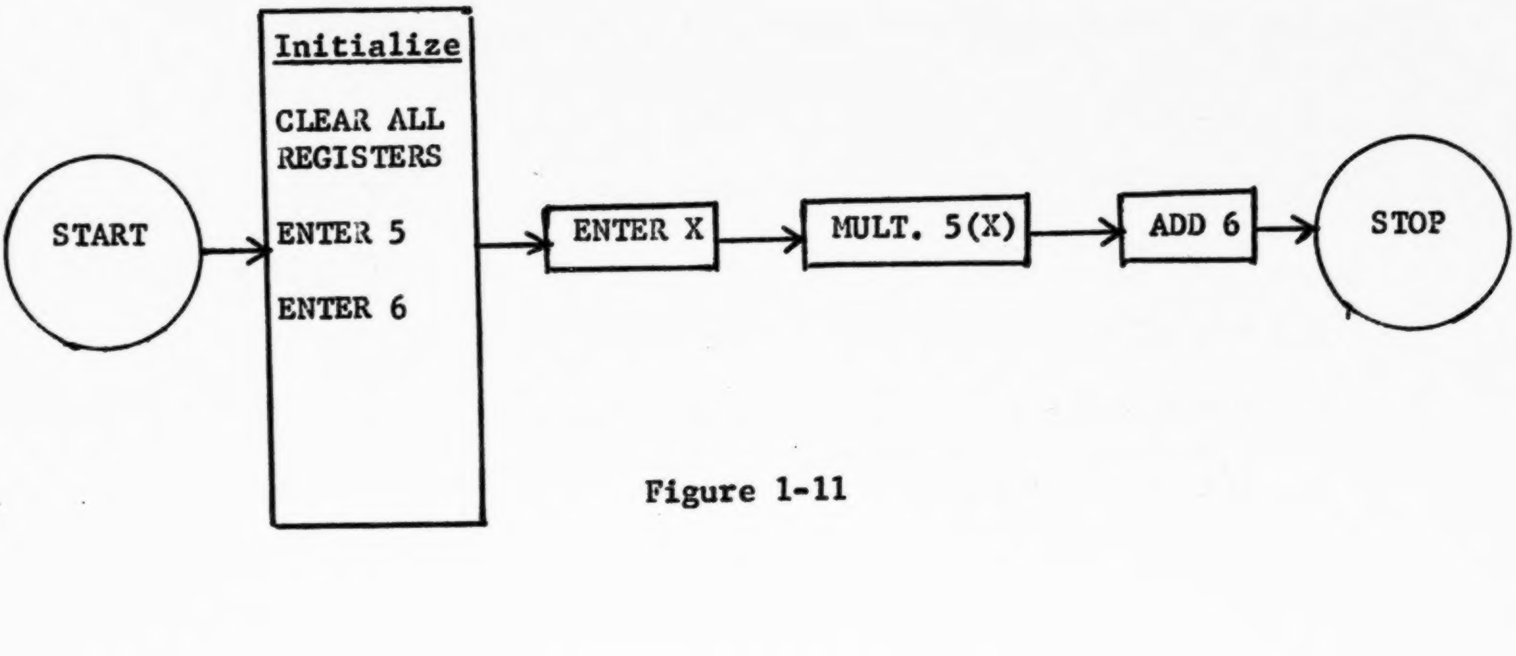


Figure 1-11

FLOW CHART FOR $c = \sqrt{a^2 + b^2}$

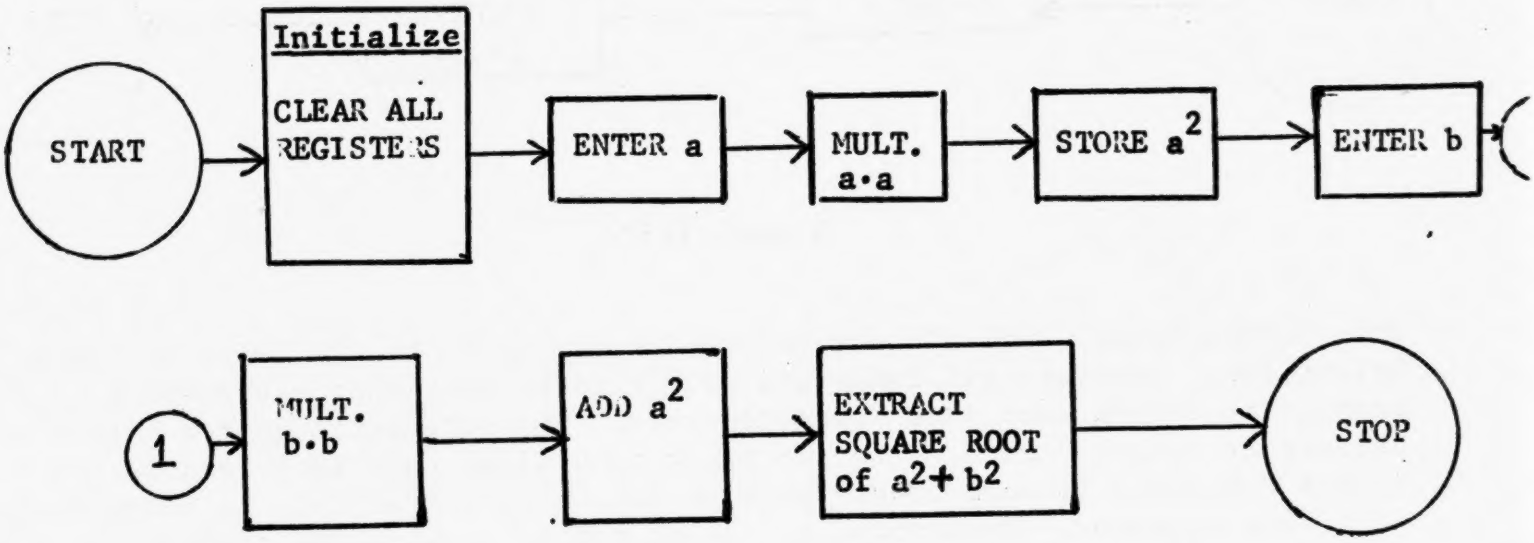


Figure 1-12

PROBLEMS

I. Draw an appropriate flow chart of the following problems.

1. $y = 5X-6$ when $X = 4$
2. $P = 2L+2W$ when $L = 4$ $W = 2$
3. $V = 4\pi r - 2\pi r$ when $r = 7$ $\pi = 22/7$
4. $C = 2\pi r$ for all r
5. $y = 3X+4X+5$ when $X = 4$
6. $\sqrt{a^2 + b^2} + \sqrt{c^2 + d^2} = X$

UNIT 2

THE COMPUTER AND OPERATIONAL PROCEDURES

Part I. Introduction

As pointed out in the note in Unit 1, Part 1, the computer is our motivational tool. Since experience shows that once the student is challenged, he will set his own rapid pace, it is imperative that he be allowed to experiment with the computer the first day. He can not hurt the computer. Therefore, gather the class around and let the students try their hand. A word or two of direction to help make them effective may be necessary. Questions flow, and the teacher becomes a resource person. All instruction, flow charts, operating procedure, and problem analysis become welcome adjuncts to a pupil and his machine. Just don't get in the way. Class work, and even self-assigned homework, will be attempted if the pupil feels it will make him a more effective operator.

Part II. Computer System

A computer is a device which accepts information, performs mathematical or logical operations with this information, and then supplies the results of these operations as new information. This process can be represented by a diagram, as follows:

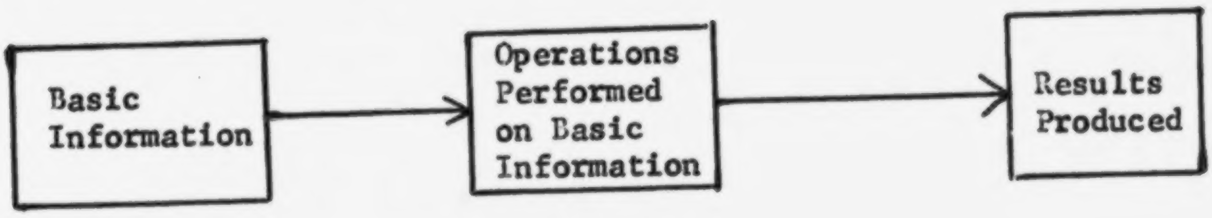


Figure 2-1

This diagram may be simplified further by expressing the three basic concepts involved as the terms input, operation, and output. They may be diagramed as follows:

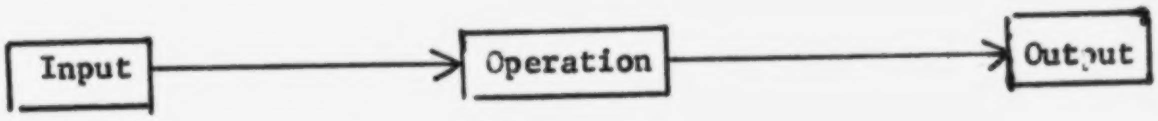


Figure 2-2

A computer system consists of the following:

1. Hardware - An array of processors and "devices" for storage or communication. An example would be the Wyle Scientific.
2. Software - Everything that is part of the system except the hardware. An example would be the Wyle cards.
3. Library - A collection of applications-oriented routines to be used as building blocks in the construction of object programs. A program for cubing a number might be found in such a collection.

The two basic types of computers are analog and digital.

1. Analog Computer

   The analog process relies on some method of continuously measuring a physical quantity, such as weight, thickness, or resistance. Examples of this process are the odometer and the speedometer. The analog computer has one major disadvantage, as compared with the digital computer, in that it cannot modify while a program is in progress; that is, each new problem must be rewired. The digital computer, however, can modify while a program is in progress.

2. Digital Computer

   In general, the digital process relies on some method of counting a sequence of discrete steps. A good example of this process is in the turnstile.

   The digital computer is physically analogous to a desk calculator, differing in only three main aspects: the computer has a memory, is faster, and can make decisions based upon predetermined criteria. Figure 2-3 can now be expanded to show all five principal computer components.

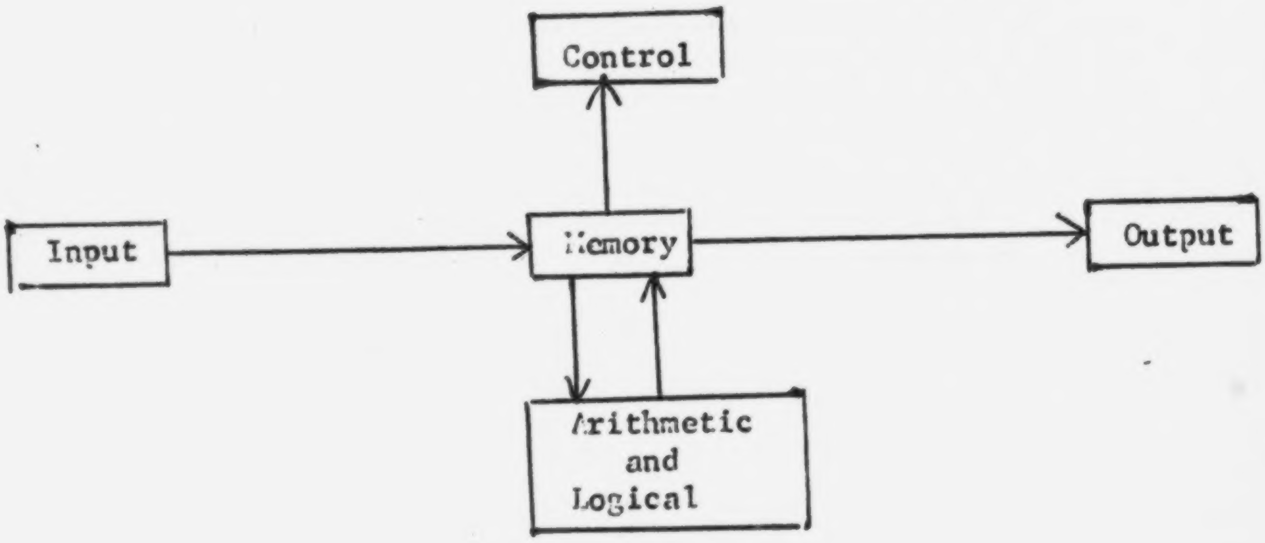


Figure 2-3

In order to gain some insight into the workings of a computer, let us follow an actual operation. The key to the operation is the "control unit." This component is capable of retrieving contents from memory in an orderly fashion. Each bit of information from memory goes to the control unit and is interpreted as a discrete instruction.

It is then executed. Executing an instruction involves calling into action one or more of the other main computer components. As an example, in performing an arithmetical calculation, the arithmetic unit and memory may be involved.

In its most simplified form, the use of a computer involves a process which places in memory a sequence of instructions in the order in which they will be selected, interpreted, and executed. A group of instructions so arranged that execution results in a certain assignment may be called a program.

Part III. Clarification of Computer Capabilities

The Desk Calculator - Given the problem of adding three numbers (call them A, B, and C) and dividing by a fourth number (call it D), the sequence of operations is as follows:

1. Press the CLEAR button. This removes from the machine any data which may have remained from previous usage.

2. Key in the first number to be added: A.

3. Press the ADD button. This causes A to be added to the number that was already in the machine--in this case, zero--and replaces the previous number with the sum ($A + 0 = A$).

4. Key in the second number to be added: B.

5. Press the ADD button. After this operation is completed, the sum, $A + B$, remains in the machine.

6. Key in the third number to be added: C.

7. Press the ADD button. This operation results in the sum, $A + B + C$.

8. Key in the divisor: D.

9. Press the DIVIDE button. This causes the number in the machine, $A + B + C$, to be divided by the new number, D, and, upon completion of the operation, leaves both the quotient and the remainder in the machine.

The Stored Program Concept - Consider the example given for the desk calculator. The sequence of operations showed nine steps or operations necessary to obtain the solution to $(A + B + C) / D$.

1. Clear

2. Enter A

3. Add A

4. Enter B

5. Add B

6. Enter C

7. Add C

8. Enter D

9. Divide by D

The tenth necessary operation is to enable the computer to stop.

10. Stop

If this were a stored-program within the computer, some of the above operations would not be necessary. These are operations 2, 4, 6, and 8 (because the values A, B, C, and D already would be stored in the memory of the computer). The program for a hypothetical computer is shown in Figure 2-4.

Program Step	Operation	Remarks
1	CLEAR	Clear accumulator register
2	ADD (A)	Add A to the accumulator
3	ADD (B)	Form A + B
4	ADD (C)	Form A + B + C
5	DIV (D)	Form (A + B + C) /D
6	STOP	Stop the computer

Figure 2-4

This same program for the Wyle Scientific is shown in Figure 2-5.

Program Step	Instruction	Remarks
1	Clear MQ, Entry, and Accumulator	
2	To Ent	
3	Stop	
4	Manually enter A	A in Ent
5	To Acc	
6	Stop	
7	Manually enter B	B in Acc
8	Add	A + B in Acc
9	Clear Ent	
10	To Entry	
11	Stop	
12	Manually enter C	C in Ent
13	Add	A + B + C in Acc
14	Clear MQ and Ent	
15	To Ent	
16	Stop	
17	Manually enter D	D in Ent
18	Divide	(A + B + C) /D in MQ
19	Stop	

Figure 2-5

Part IV. Characteristics of the Wyle Scientific

Display (See Figure 2-6.)

1. The top line of the display is the Multiplier - Quotient Register, abbreviated MQ. This register is used to hold the multiplier in multiplication operations, the quotient in division operations, and the answer in square root operations.

2. Underneath the MQ register is the Entry register, abbreviated Ent. This register holds the multiplicand in multiplication operations, the divisor in division operations, and the minuend in subtraction operations. It is the normal register for data entry, and its contents are added to the

accumulator register in the normal addition operation.

3. Underneath the Ent register is the accumulator register, abbreviated Acc. This register is used to hold the original number (radicand) in square root operations, the dividend in division operations, the subtrahend in subtraction, and the product in multiplication.

4. Underneath the Acc register are the three storage registers numbered 1, 2 and 3 from top to bottom. These registers are abbreviated R1, R2 and R3. They are used to store constants or intermediate answers which may be required at a later stage in the calculation. Numbers can be transferred to any of these registers from any other register and from any of these registers to any other register.

5. An indicator zero appears on the far left; in Figure 2-6 it is shown aligned with the MQ Register. This indicates which register has been selected as the "From" register. In one register, one of the 24 digits will be intensified. This is the register selected as the "To" register. When the transfer key is depressed, the contents of the selected "From" register will automatically be transferred to the "To" register. In the illustration, this is shown as the initial zero of the Acc. register.

MQ Register	0	000	000	000	001	.	414	213	562	373
Entry Register		000	000	000	000	.	000	000	000	000
ACC. Register		000	000	000	000	.	000	000	000	000
Storage Reg. 1		000	000	000	000	.	000	000	000	000
Storage Reg. 2		000	000	000	000	.	000	000	000	000
Storage Reg. 3		000	000	000	000	.	000	000	000	000

Wyle SCIENTIFIC Visual Display

Figure 2-6

Clearing the Registers - The three arithmetic registers (MQ, Entry, Acc) are cleared; that is, the contents are eliminated by means of the three keys located in the lower right hand section of the keyboard. These are labeled:

Clear Entry
Clear MQ
Clear Acc

and each key eliminates all data in the corresponding register. When a register is cleared, it is automatically addressed "To."

To clear Registers 1, 2 and 3, "Transfer To" the selected register from some other register which is already cleared.

Data Entry - Numerical data is entered via the numerical keys located in the center of the keyboard. Data will appear in the selected "To" register. The "To" register is identified by the "To" marker, and the position of this marker indicates the position where the next digit will be entered.

Numbers are entered exactly as read, including the decimal point. As an example, the following steps:

Depress To Acc
4 Key
3 Key
5 Key
Decimal point (1) Key
0 Key
1 Key
4 Key

will place 435.014 in the selected "To" register, properly aligned about the preselected decimal point.

Forward and Back Space - The "Forward Space" and "Back Space" keys, located in the top center section of the keyboard, position the "To" marker one digit at a time. This enables you to correct an erroneous entry without re-entering the data.

Shift Left and Shift Right - These keys move the entire number in the selected "To" register to either the left or right, one space each time the keys are depressed.

Decimal Point - The Wyle Scientific handles the decimal point automatically; however, as a convenience, the decimal point may be positioned in steps of three digits. This permits calculations using numbers through the range between a 3-digit whole number with a 21-digit fraction and a 21-digit whole number with a 3-digit fraction.

Mathematical Operation

1. Addition

When the add (+) key is depressed with the "Add Any Register" switch off (down), the contents of the Entry register are added to the contents of the Acc register. The sum appears in the Acc display.

In symbolic notation:

$$(\text{Entry}) + (\text{Acc}) \longrightarrow \text{Acc}$$

2. Subtraction

When the Sub key is depressed with the "Add Any Register" switch off (down):

$$(\text{Acc}) - (\text{Entry}) \longrightarrow \text{Acc}$$

3. Clear and Multiply

Depressing this key first clears the Acc to zero and then:

$$(\text{MQ}) \times (\text{Entry}) \longleftrightarrow \text{Acc}$$

4. Multiply and Add

Depressing the (Mult +) key results in:

$$(\mathrm{Acc}) + \left|(\mathrm{MQ}) \times (\mathrm{Entry})\right| \longrightarrow \mathrm{Acc}$$

5. Multiply and Subtract

Depressing the (Mult -) key results in:

$$(\mathrm{Acc} - \left|(\mathrm{MQ}) \times (\mathrm{Entry})\right| \longrightarrow \mathrm{Acc}$$

6. Divide

Depressing the ($\div$) key results in:

$$(\mathrm{Acc}) \div (\mathrm{Entry}) \longrightarrow \mathrm{MQ}$$

7. Square Root

Depressing the ($\sqrt{}$) key results in:

$$\sqrt{\mathrm{Acc}} \longrightarrow \mathrm{MQ}$$

Transfer

Depressing the "Transfer" key will transfer the contents of the selected "From" register to the selected "To" register (See Part IV- 4).

Special Operations

Part IV of this unit has, up to this point, described those operations which must be mastered to use the Wyle Scientific with a degree of competence. The following describes operations which are useful in more complex problems and which extend appreciably the capabilities of the machine.

1. Add from any Register

When this switch is in the "on" (up) position, the contents of the selected "From" register may be added to or subtracted from the contents of the Acc register. In the "off" (down) position, only the contents of the Entry register can be added to or subtracted from the contents of the Acc register.

Example 1

Clear all Registers
Place Add (from) Any Reg. key in "on" (up) position

Depress	To Acc
Enter	25
Depress	To R1
Enter	18
Depress	From R1
	Add

25 + 18 = 43 appears in Acc

Note: Contents of R1 added to contents of Acc

$$(\mathrm{R1}) + (\mathrm{Acc}) \longrightarrow \mathrm{Acc}$$

The number being added is not erased in this mode of operation; 18 would remain in R1.

2. OVERFLOW (See Figure 2-7.)

When the "Overflow Lock Off" Switch is in the "off" (up) position, the overflow lockout is inhibited. Overflow normally occurs when the answer to an operation exceeds the capacity of the machine, as for example, when two seven-digit numbers are multiplied together with the decimal in the center position (only 12 digits available for product). Overflow indication consists of all digits in the display intensified, plus a line of zeros at the far right edge of the display. Overflow may be unlocked by depressing the TO ACC (or any "TO" key). The problem may be repeated after moving the decimal point appropriately.

Example 2

Place overflow lock down.

Multiply 7 863 571. by 8 436 211. with decimal point set at middle position. Overflow occurs. The correct answer is 66 338 744 169 481, obtained by moving the decimal point 3 places to the right and repeating the operation. With the decimal point in the center position and overflow lock off (up), repeat the multiplication. The answer in the accumulator is incorrect.

This exercise shows that the overflow lock should be "on" (down) when performing large number operations. It should be "off" only under conditions shown in Example 2A.

Perform Example 2A with overflow lock "ON" (down).

Example 2A

Clear MQ, ENTRY and ACCUMULATOR

Depress	TO ENTRY
Enter	15.
Depress	FROM ENTRY
	ADD
	CLEAR ENTRY
Enter	16.
Depress	SUB

The accumulator contains the ten complement of the Answer - 1.

Depress	CLEAR ENTRY
Enter	2.
Depress	ADD

The machine overflows, since we were adding 2. to 999 999 999 999.

MQ Register	0	000	007	863	571	.	000	000	000	000	0
Entry Register		863	571	000	000	.	000	000	000	000	0
ACC. Register		000	000	000	000	.	000	000	000	000	0
Storage Reg. 1		000	000	000	000	.	000	000	000	000	0
Storage Reg. 2		000	000	000	000	.	000	000	000	000	0
Storage Reg. 3		000	000	000	000	.	000	000	000	000	0

Example 2: Overflow indication

MQ Register	0	000	000	000	000	.	000	000	000	000	0
Entry Register		000	000	000	002	.	000	000	000	000	0
ACC. Register		000	000	000	001	.	000	000	000	000	0
Storage Reg. 1		000	000	000	000	.	000	000	000	000	0
Storage Reg. 2		000	000	000	000	.	000	000	000	000	0
Storage Reg. 3		000	000	000	000	.	000	000	000	000	0

Example 2A: Overflow on adding 2. to 999 999 999 999
(the tens complement of - 1)

Figure 2-7

Repeat example 2A with overflow lock "off" (up). The correct answer (- 1. + 2. = 1.) appears in the accumulator.

The example illustrates that the overflow lock, when off, permits ADD and SUB, and both negative and positive cumulative multiplication operation in the negative number region, the answer being in true form if it is negative.

3. Complementing

When the answer is in complement form (negative), the re-complementing operation is as follows:

Perform example 2A up to the first subtraction; then, continue as in example 3.

Example 3

Depress	From Acc
	To Entry
	Transfer
	Clear Acc
	From Entry
	Sub

The correct answer 1. appears in the Acc. It should be remembered that it is negative.

4. Keep Remainders

When the KEEP REMNDR switch is on (up), both the divisor and the remainder are displayed after all division operations. The divisor stays in the Entry register but is shifted left so that the first digit of the divisor is one place to the left of the first digit of the quotient. The remainder appears in the Acc register and is shifted left one more time than the divisor is shifted. Also, twice the root is retained in the Entry register after a square root operation, and 10 times the true remainder is retained in the Acc register. For example, with KEEP REMNDR "on", perform Example 4.

Example 4

Depress	Clear Acc
Depress	To Acc
Enter	144
Depress	To Entry
Enter	13
Depress	Divide (÷)

The quotient 11.076 923 076 923 appears in the MQ register. The divisor is in the Entry register but is shifted left one place so that the first digit is one place to the left of the first digit of the quotient. The divisor therefore appears as 130. The remainder of 1. is in the Acc register but is shifted left two places (one more than the divisor is shifted) and appears as 100.

Depress	To Acc
Depress	Shift Right
Depress	Divide (÷)

The digits to the right of the decimal are additional digits of the quotient. This operation may be continued indefinitely for any precision of division that may be required.

5. Punched Card Programing

Programs to be executed are punched on one or more 40-column cards of the type shown below. This is a conventional card, over-printed with a pattern which identifies the various columns. The PC-01 reads a row at a time,and there are 39 possible punch positions in each row. The 40th column is a strobe column (located in the center of the card) and is punched in all rows.

Thirty-eight of the columns correspond to the 38 keys on the SCIENTIFIC keyboard. A punch in the far right hand column, the STOP column, causes the PC-01 to stop reading so that data or instructions can be entered manually from the keyboard. Holes are punched in the card in the same sequence as the manual keyboard would be operated to accomplish the same task.

Cards are pre-scored and can be punched with a simple stylus or a ball point pen. Unscored cards are also available and cards can be duplicated on conventional keypunch equipment.

If a program requires more than one card, as is usually the case, cards can be taped together, edge to edge, with black tape. Transparent, or semi-opaque tape should not be used for this purpose since the photo-electric reading circuits may react to light passed through the tape.

Sample Program

The card shown in Figure 2-8 is punched with the program to compute a, where $a^2 = b^2 + c^2$.

Step No.	Instruction	Notes
1	CLEAR MQ- CLEAR ENTRY	Two CLEAR instructions may be punched on a single row. Up to 3 registers may be addressed TO in a single row.
2	TO MQ- TO ENTRY	
3	STOP	Manually enter b.

Step No.	Instruction	Notes
4	CLEAR AND MULTIPLY	$b^2 \longrightarrow$ Acc
5	CLEAR MQ	
6	TO MQ- TO ENTRY	
7	STOP	Manually enter c.
8	MULTIPLY AND ADD	$b^2 + c^2 \longrightarrow$ Acc
9	SQUARE ROOT (⌐)	$\sqrt{b^2 + c^2} \longrightarrow$ MQ a
10	STOP	Answer appears in MQ register

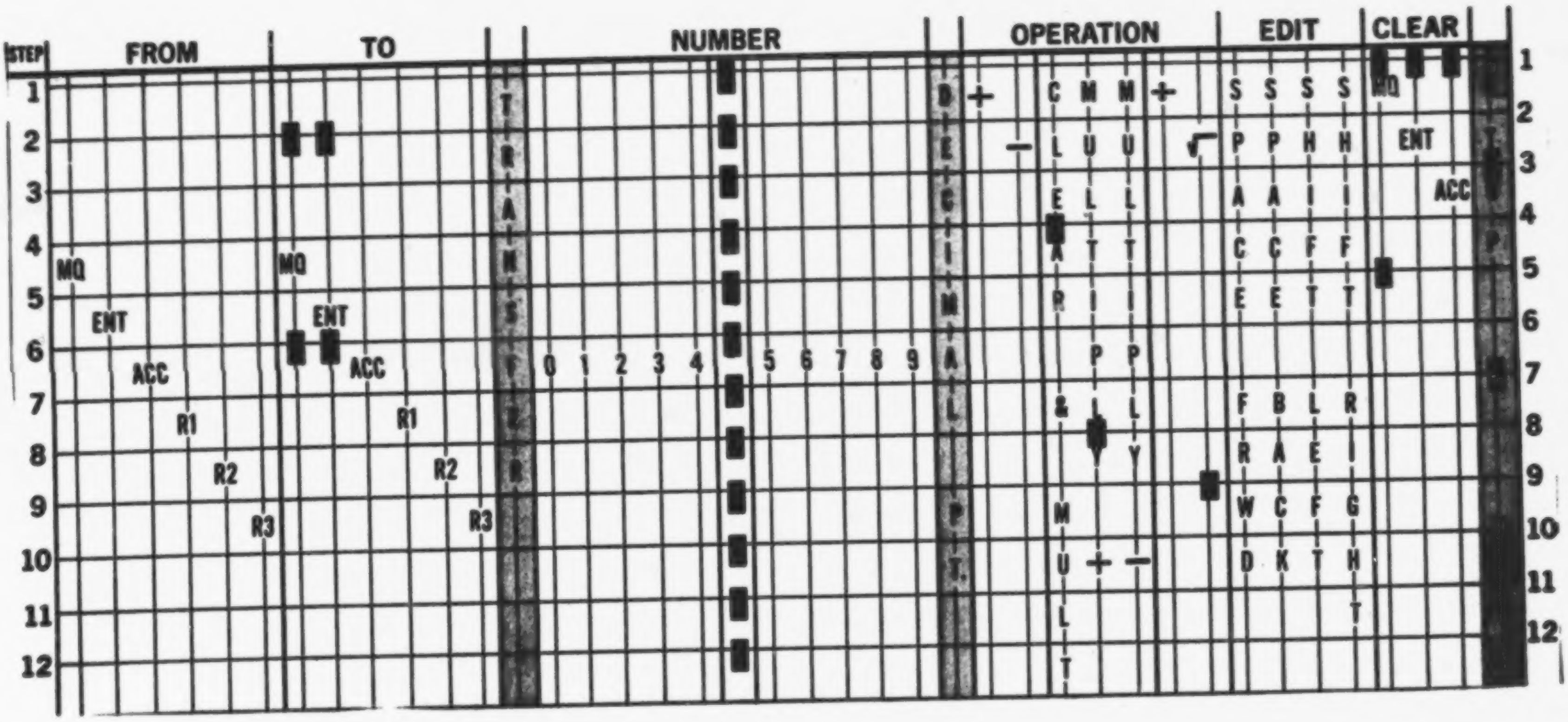


Figure 2-8

6. Programing Restrictions

A. FROM any register, TO a maximum of three registers, and TRANSFER may be punched on the same line on the Wyle card, providing that no more than two of these registers are storage registers.

B. In the "Add Any Register" mode, both the "From" address and the "Add" (or Sub) command may be punched in a single row.

C. All three arithmetic registers can be cleared on the same line on the Wyle card.

D. Enter to a maximum of three registers, on the same line on the Wyle card, providing no more than two of the registers are storage registers.

E. Both the "To" and "From" addresses and the "Transfer" command can be punched in a single row.

7. Problem Solving

To solve a problem with a computer system requires, in roughly chronological order, the following steps:

1. Statement of the problem
2. Organization of the problem including a flow chart
3. Detailed design and coding of routines
4. "Debugging" (testing and correcting routine)
5. Execution of the program

These steps can best be explained by programing a simple problem on the Wyle computer.

Example

Program $y = 5x + 6$ on the Wyle Scientific

Statement of the Problem

Program $y = 5x + 6$

Organization and Flow Chart

1. Initialize
2. Enter variable
3. Form 5 x
4. Form $5x + 6$
5. Stop

Flow Chart

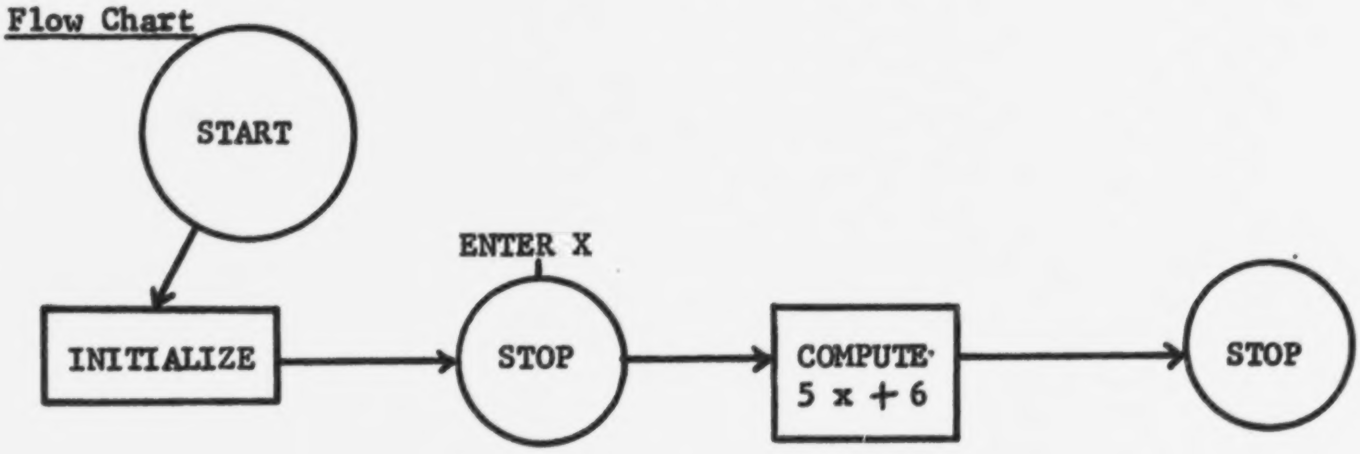


Figure 2-9

Note: See Figure 1-2 for an explanation of flow chart symbols used in Figure 2-9. Initialize includes clearing of all registers, depressing To MQ key and entering 5 in MQ, depressing To Entry key.

Coding of Routine	Comments
1. Clear MQ, Entry, Acc, R1, R2, R3	
2. To MQ - enter 5	5 in MQ
3. To Entry	
4. Stop - enter variable x	x in Entry
5. To Acc - enter 6	6 in Acc.
6. Multiply and Add	5 x + 6 in Acc.
7. Stop	

Debugging

Put any value of X into the equation (say X = 3); "program" into the computer; and the answer, in the Acc., should be 21. If it is, then the problem is probably programed correctly.

Execution of the Program

Punch the 7 steps shown under "coding of routine" onto Wyle program card in exact sequence as displayed. Each step is to be punched on a separate line of the Wyle program card.

The "Coding of routine" could be done in different ways.

PROBLEMS

Chart the following problems in detail and program on the Wyle card.

1. $A + B - C$
2. $\frac{A}{2} + B \cdot C$
3. $\sqrt{A + B}$
4. $\frac{A \cdot B + C}{3}$
5. $Y = 9X$ for all values of X

UNIT 3

LOOP PROCESSES

Part I. Looping - The Single-Loop Process

Looping is basic to any study of programing and coding. To get useful service from a computer, it usually is necessary to arrange for groups of instructions to be executed repeatedly. The most common technique for doing this is known as looping. Figure 3-1 is a simple example of the single-loop process.

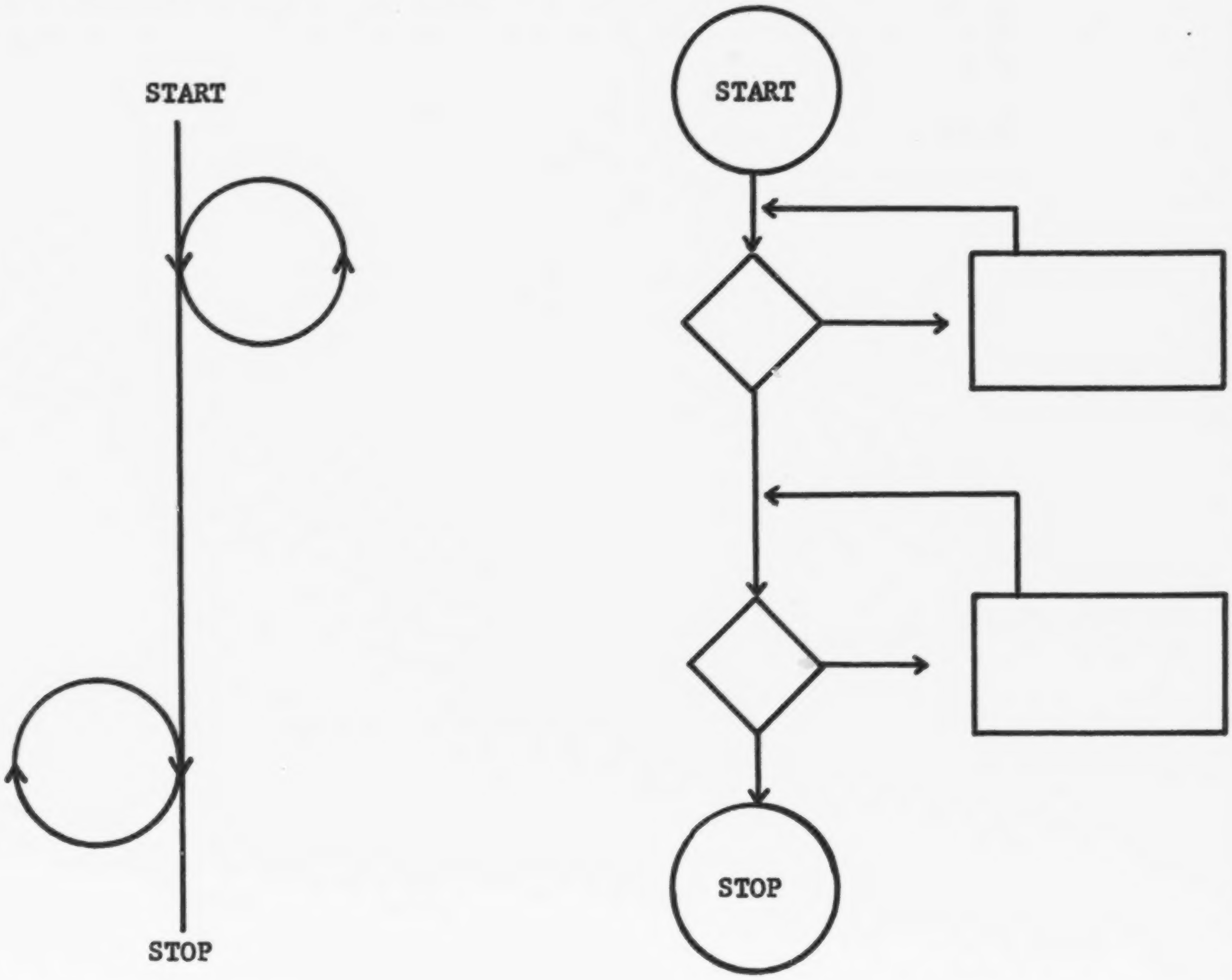


Figure 3-1. Single-Loop Process

Note that the sequence is interrupted by a loop and that certain steps are repeated; then, the sequence is continued; it is interrupted again by another loop; and, finally, it proceeds to the stopping point.

The most obvious advantages of looping are:

1. The writing of instructions does not involve repetitive, monotonous work. Looping provides for repetition.

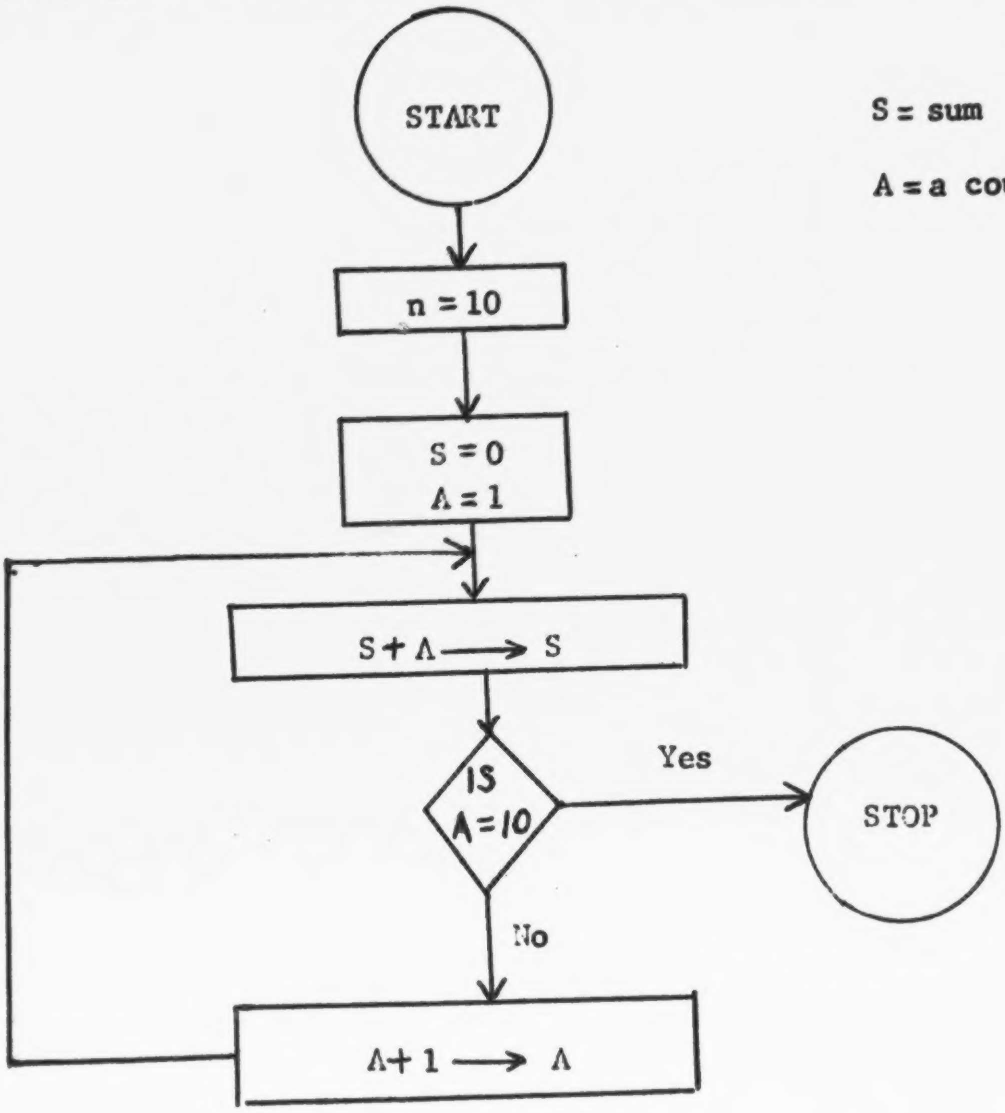
2. Debugging is made relatively easy. In general, if any digit in the instructions is wrong, the whole loop will collapse, usually making it easy to detect coding errors.

3. The loop can readily be modified to fit changing conditions.

In order to reduce the chances for error in coding, the following check should be made to assure that a loop is correctly written; at least, for the number of times it will be executed. Pretend that you wrote it to do the job just once. Then, determine the change necessary in the test to do the job n times. If it still looks correct, the chances are excellent that it will work.

Example 1

Find the sum of the first ten counting numbers.



S = sum

A = a counting number

Figure 3-2

Counting Cycles in a Loop

You will notice in Figure 3-2 that we have let the computer do the heavy tedious work for us. By initializing $n = 10$, $A = 1$ and building in a loop we do not have to write program instructions for each of the additions performed. Step through the loop a few times and you will see how the sum is increasing, the counter A is keeping track of the number of cycles, and the decision instruction allows us to leave the loop at the appropriate time. If n were 1000 imagine how much effort this would save us. Actually, all computers have a finite amount of storage and without looping, we would exhaust storage just trying to put in our program.

The sum of the first n integers $S = \frac{n}{2}(n+1)$, might be introduced here to point out that some problems are better done by thinking pupils rather than by computers.

Example 2

Find the greatest common divisor (G.C.D.) or the highest common factor (H.C.F.) of two integers.

Euclid is responsible for this algorithm, which involves repeated division. The steps in this algorithm are as follows:

1. Divide the large integer, I_1, by the smaller integer, I_2, and call the remainder R_1.
2. Divide I_2 by R_1; the remainder being R_2.
3. Divide R_1 by R_2; the remainder being R_3.
4. Continue this process until any remainder equals zero ($R_n=0$).

The last divisor used is the greatest common divisor. To find the greatest common divisor of 15 and 25, using Euclid's algorithm:

$$I_1 = 25,\ I_2 = 15$$

$$\begin{array}{r|l} & 1 \\ \hline 15 & 25 \\ & \underline{15} \\ & 10 \end{array}$$

$R_1 = 10$

$$\begin{array}{r|l} & 1 \\ \hline 10 & 15 \\ & \underline{10} \\ & 5 \end{array}$$

$R_2 = 5$

$$\begin{array}{r|l} & 2 \\ \hline 5 & 10 \\ & \underline{10} \\ & 0 \end{array}$$

Since 5 is the last divisor used for which $R_n = 0$, the greatest common divisor of 15 and 25 is 5.

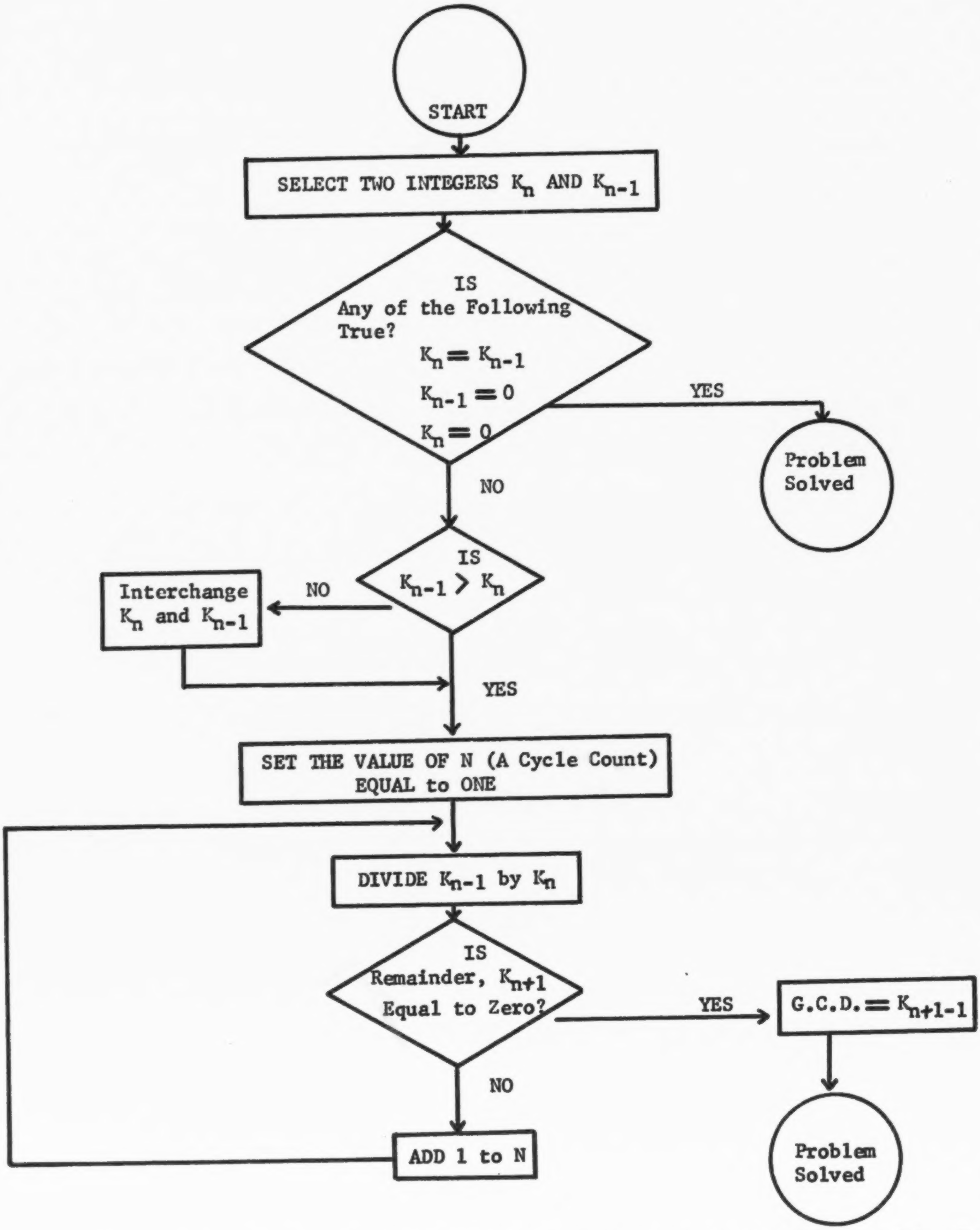
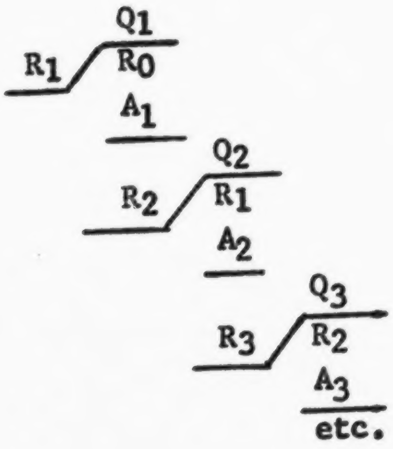


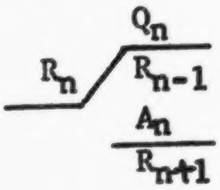
Figure 3-3. Finding the Greatest Common Divisor of Two Integers -- Using Words

The cycle count referred to in Figure 3-3 can best be explained as follows:

The algorithm for finding the greatest common divisor is:



Each cycle of this algorithm can be represented as:



where n represents the number of the cycle and increases as the process continues. When the remainder, R_{n+1}, becomes zero, the greatest common divisor has been found, and it is $R_{n+1-1} = R_n$

There is a very simple and clever flow chart for Example 2. This is shown in Figure 3-4.

However three new symbols must be introduced, in order to understand this flow chart. These symbols are as follows:

1. $B \leftarrow B - A$ means that the value of B takes the value of B - A.

2. $B \leftrightarrow A$ means the values of A and B are interchanged.

3. $[\![X]\!]$ denotes the greatest integer n such that $n \leq X$.

   To find $[\![X]\!]$, therefore, we "round off" the decimal representation of X to the next lowest whole number.

   Thus: $[\![\pi]\!] = 3$, $[\![1.7]\!] = 1$, $[\![-\frac{1}{2}]\!] = -1$

The relative simplicity and beauty of the flow chart shown in Figure 3-4 is obvious when it is compared to the flow chart of this same problem, as shown in Figure 3-3, using words instead of symbols.

Figure 3-4, thus, is an excellent example of how a complicated flow chart, as shown in Figure 3-3, can be simplified by use of a different approach to the problem, as well as by using proper symbols.

Examples of single-loop flow charts follow.

Let A and B represent any two integers.

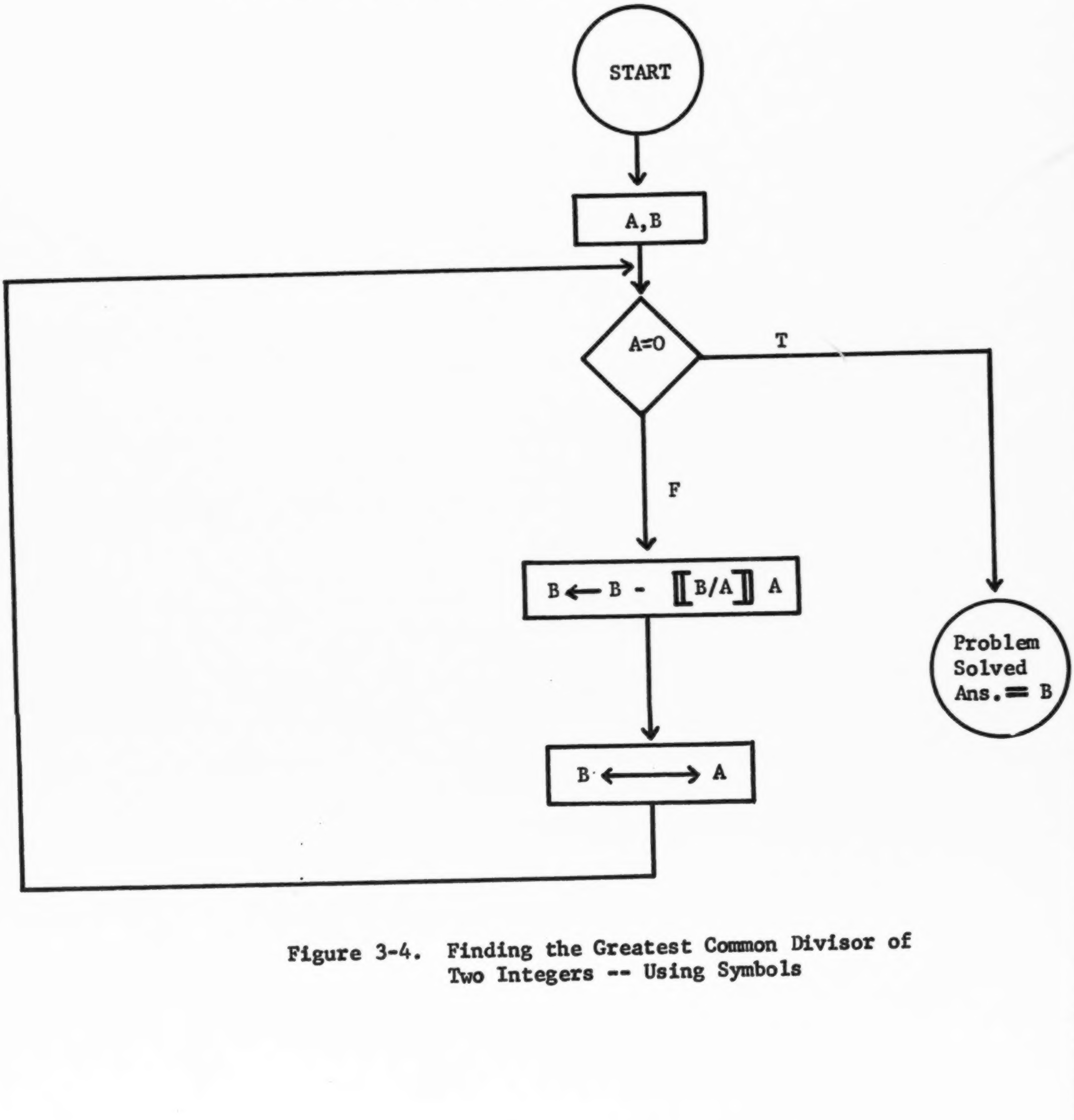


Figure 3-4. Finding the Greatest Common Divisor of Two Integers -- Using Symbols

Example 3 Draw a flow chart showing multiplication by repeated addition.

One way to multiply one nonzero number by another is to add successively the multiplicand and subtract 1 from the multiplier for each addition. When the result of the subtraction sequence becomes zero, the process is finished and the product is equal to the final sum. For example, multiply 15 by 5.

Multiplicand = 15

Step number (N)		Multiplier Difference
0	= 00	
1	0+15 = 15	5 - 1 = 4
2	15+15 = 30	4 - 1 = 3
3	30+15 = 45	3 - 1 = 2
4	45+15 = 60	2 - 1 = 1
5	60+15 = 75	1 - 1 = 0

Step 5 shows the subtraction sequence equal to zero; therefore, the final sum (75) is the answer.

NOTE: The cycle count in Example 3 would be N = 5.

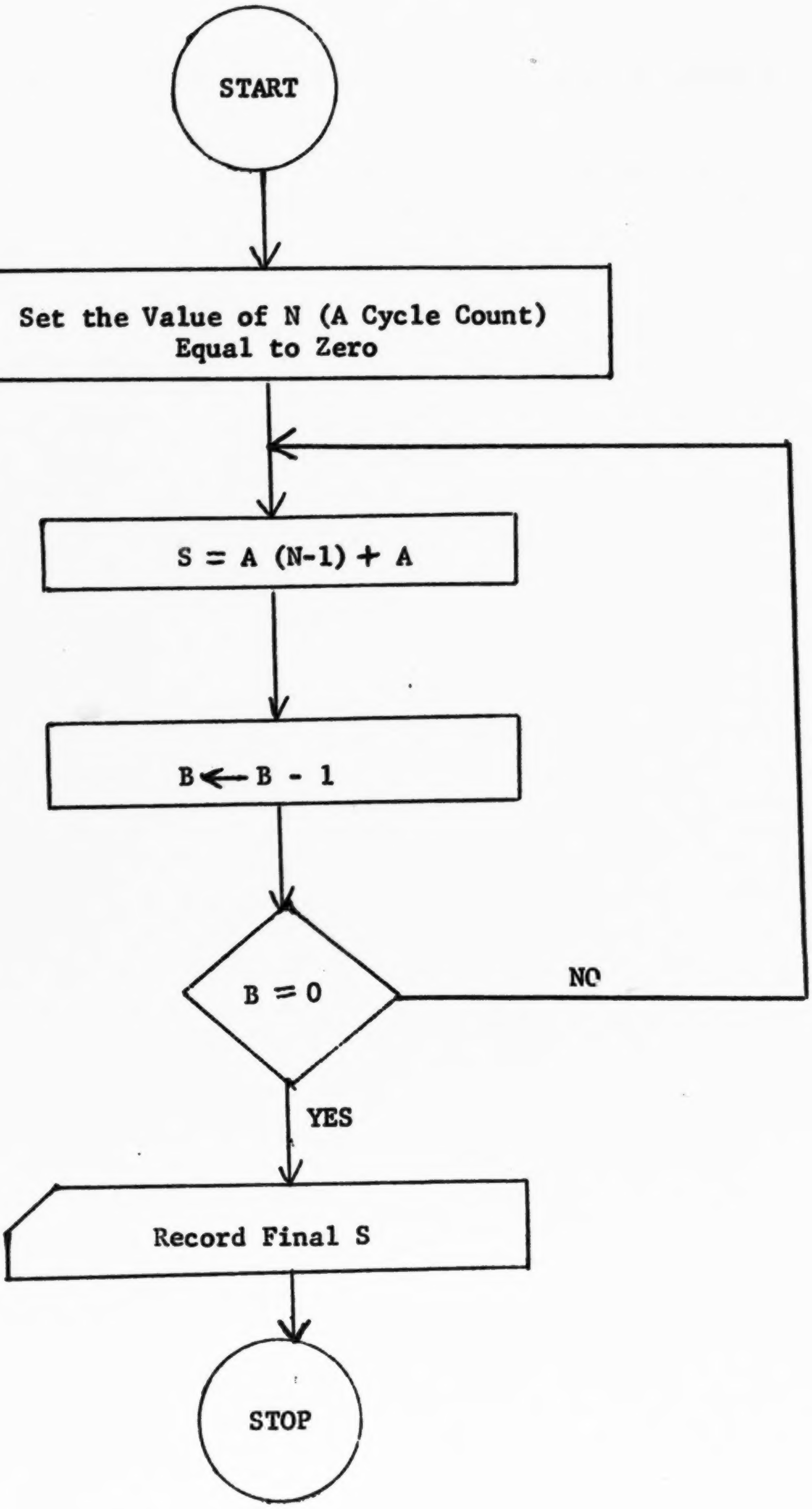
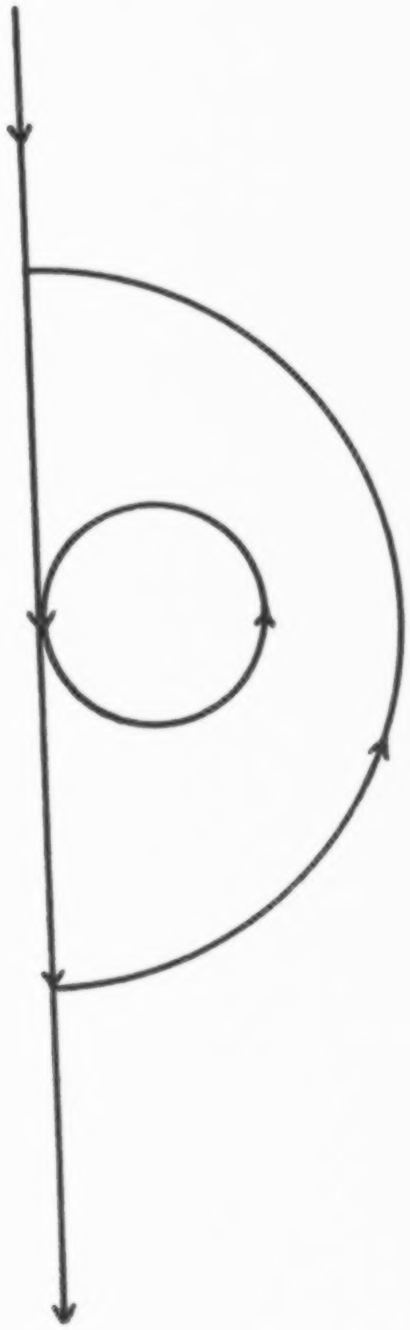


Figure 3-5. Multiplication by Repeated Addition

Part II. The Nested-Loop Process

The nested loop is distinguished from the single loop by the fact that it has a loop within a loop. That part of the sequence which is repeated is again repeated in its entirety. Figure 3-5 is a diagram of this process.



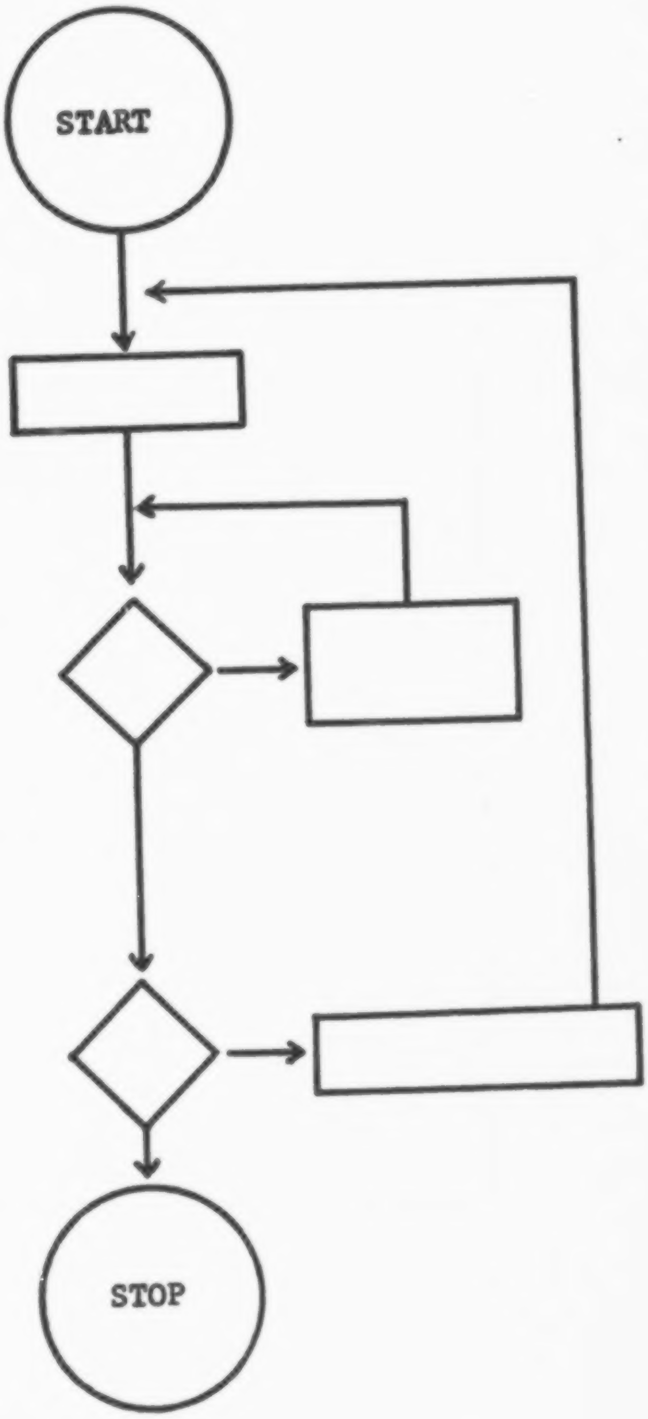


Figure 3-6. Nested-Loop Process

A good example of nested looping is shown in Figure 3-6.

Example 4

Find all the prime factors of an integer.

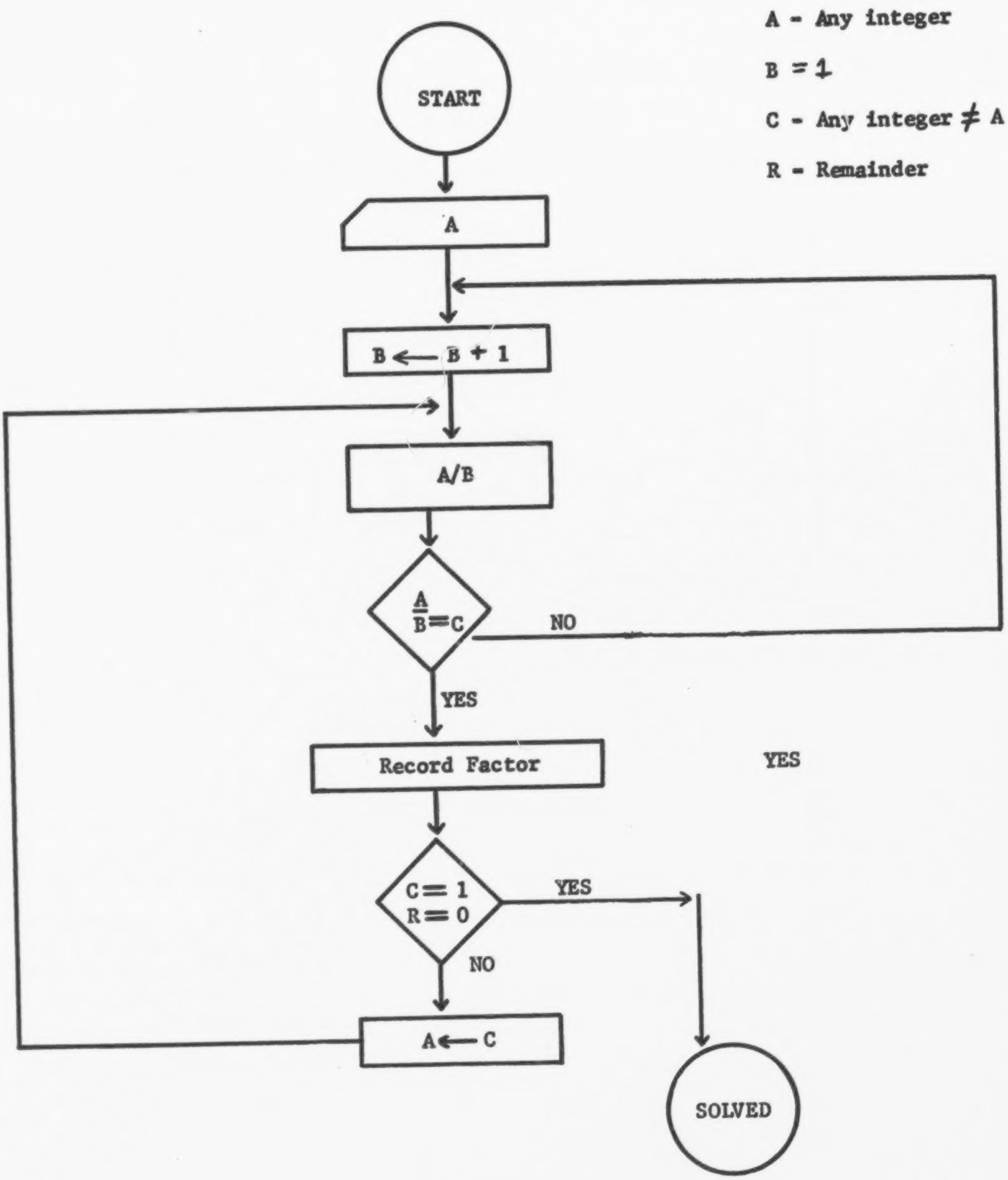


Figure 3-7. Prime Factors of an Integer

In Figure 3-7, when the first loop is completed, it then is programed through the entire flow diagram again. This makes it a nested loop.

To check the accuracy of the flow chart shown in Figure 3-7. let A, the integer, be equal to 30. The problem, then, is to find all the prime factors of 30.

Note that for a given integer, the integer itself and 1 are not considered factors.

1. Try 2 as a factor.
   $A = 30$ $B = 2$ $A/B = 15 = C$. Since C is an integer not equal to A, we record the factor 2. Since $C \neq 1$, we put C in place of A. Therefore, $A = 15$.

2. On this first loop, $A = 15$; B (next number) 3.
   $A/B = 15/3 = 5 = C$. Again, since C is an integer $\neq$ A, record the factor 3. $C \neq 1$, so we again put C in place of A. Therefore, $A = 5$.

3. On the second loop, $A = 5$; B (next number) $= 4$.
   $A/B = 5/4 \neq$ integer and we loop back to beginning. The reader should continue through the problem until it is completed.

To check the accuracy of the loop on the right side of the flow chart, let $A = 15$.

1. Try the first (next) prime, 2, as a factor.
   $A = 15$ $B = 2$ $A/B = 7\frac{1}{2} = C$.

2. Since C is not an integer, and since $C \neq 1$, we take the first loop.

3. B (next prime) $= 3$; $A = 15$ $A/B = 5$. Therefore, 3 will be recorded.

4. Continuing this process, 5 also will be recorded. This process can be diagramed as follows:

$2\underline{/30}$	$3\underline{/15}$
$3\underline{/15}$	$5\underline{/5}$ 1
$5\underline{/5}$ 1	

Therefore, $30 = 2 \cdot 3 \cdot 5$ $15 = 3 \cdot 5$

Example 5

Find the greatest common divisor (G.C.D.) or the highest common factor (H.C.F.) of two integers.

Note that this is exactly the same problem as Example 2. However Example 2 was solved using a single loop. The solution shown in Figure 3-8 has a loop within a loop, or a nested loop.

To check the accuracy of the flow chart shown in Figure 3-8, the equality $A = B$ and both inequalities $A > B$ and $A < B$ should be used.

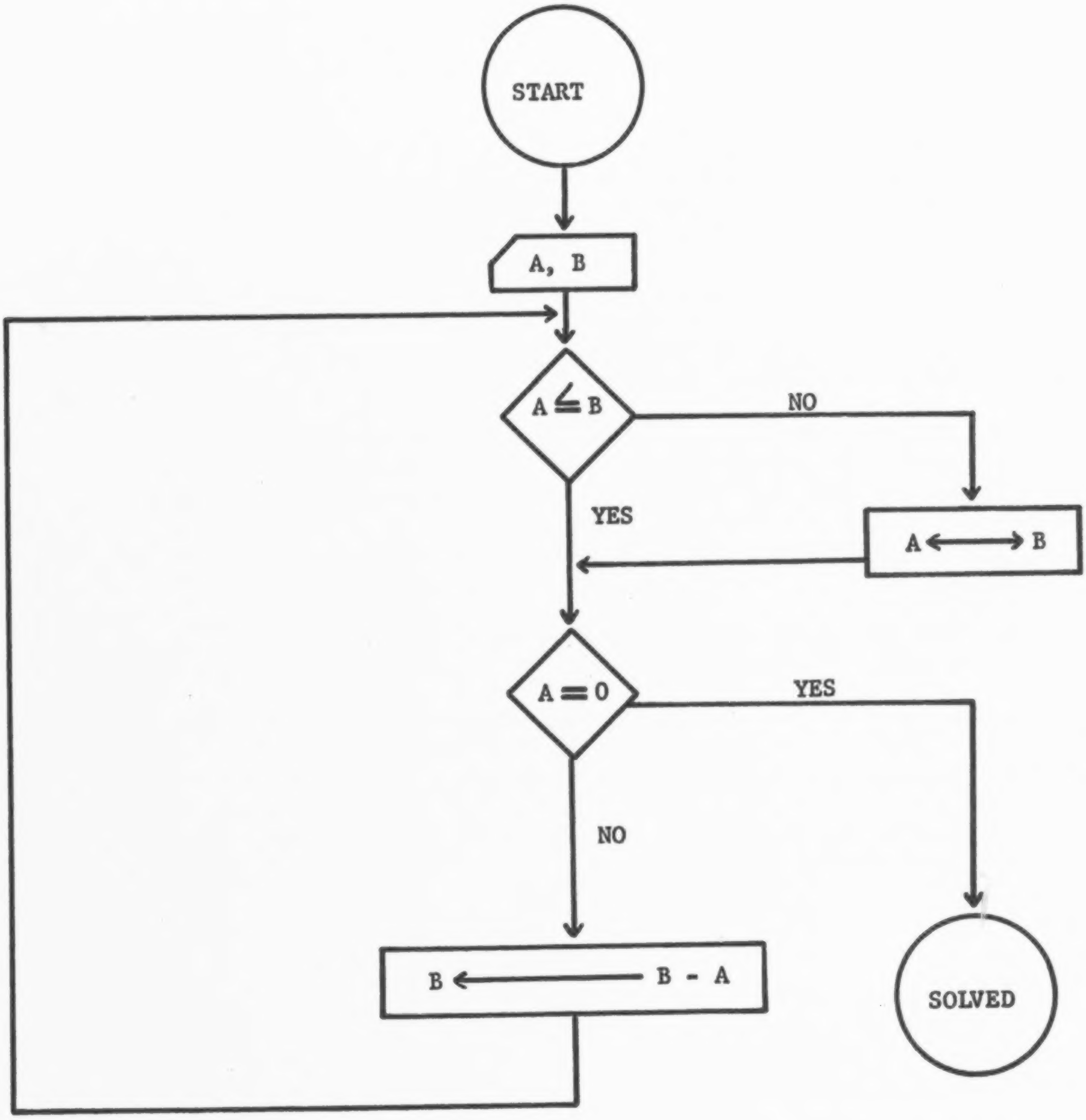


Figure 3-8 Greatest Common Divisor

or

Highest Common Factor

IF $A = B$, Let $A = 20$ $B = 20$

$A \neq 0$

$B \longleftarrow B - A$

$B - A = 0$

$B = 0$

On first loop, $A = 20$, $B = 0$

$A \not< B$

$A \longleftrightarrow B$

$A = 0$, $B = 20$

Answer $= 20$

If $A < B$, Let $A = 20$, $B = 30$

$A \neq 0$

$B \longleftarrow B - A$

$B - A = 10$

$B = 10$

On first loop, $A = 20$, $B = 10$

$A \not< B$

$A \longleftrightarrow B$

$A = 10$, $B = 20$

$A \neq 0$

$B \leftarrow B - A$

$B - A = 10$

$B = 10$

On second loop, $A = 10$, $B = 10$

$A = B$

$A \neq 0$

$B \leftarrow B - A$

$B - A = 0$

$B = 0$

On third loop, $A = 10$, $B = 0$

$A \nless B$

$A \leftrightarrow B$

$A = 0$, $b = 10$

Answer $= 10$

If $A > B$ Let $A = 25$, $B = 10$

$A \nless B$

$A \leftrightarrow B$

$A = 10$, $B = 25$

$A \neq 0$

$B \longleftrightarrow B - A$

$B - A = 15$

$B = 15$

On first loop, $A = 10$, $B = 15$

$A < B$

$A \neq 0$

$B \leftarrow B - A$

$B - A = 5$

$B = 5$

On second loop, $A = 10$, $B = 5$

$A \nless B$

$A \leftrightarrow B$

$A = 5$, $B = 10$

$A \neq 0$

$B \longleftarrow B - A$

$B - A = 5$

$B = 5$

On third loop, $A = 5$, $B = 5$

$A = B$

$A \neq 0$

$B \leftarrow B - A$

$B - A = 0$

$B = 0$

On fourth loop, $A = 5$, $B = 0$

$A \not< B$

$A \leftrightarrow B$

$A = 0$, $B = 5$

Answer $= 5$

PROBLEMS

1. Draw a flow chart for the following algorithms:

   A. Division by repeated subtraction

   B. Greatest common divisor of more than two integers

2. Plan the use of the Wyle computer with a unit of work. Use the following format in your lesson plan:

   A. Organize the problem. Provide for the most economic use of time and program cards.

   B. Plan the approach. Prepare specific problems to be presented to the pupils.

   C. Code the problem. Use the Wyle Coding Sheet. Remarks are very important; for example, it might be noted that the Wyle Computer does not handle negative numbers _per se_.

   D. Test-check with a simple test case.

In the lesson plan, include involvement of all pupils. Consider what other class members will be doing while the computer is being used by only a few of the pupils.

UNIT 4

DEVELOPING EXAMPLES AND METHODS TO OBTAIN "OBJECTIVES OF COMPUTER PROGRAM"

Choosing the type of computer to be used is of extreme importance. We must constantly keep our sights on the goals. This is not to be a course in teaching programing. The computer is to be used as an aid in teaching mathematics. Consequently, there are certain things we do not want the computer to do. For example, the Wyle Scientific cannot handle negative numbers. This forces the pupil to learn about complements. Furthermore, this computer does not give the complement. The pupil must perform the operation of subtraction in order to obtain the desired result. Actually, the most elementary type of computer is the most effective for our purposes. Three arithmetic registers and three storage registers have proved very satisfactory.

Since the Wyle has only limited storage and limited capabilities, the student will be able to develop facility in handling problems. An equally important facet of these limitations is that every problem must be carefully analyzed and understood by the student as he prepares his program. If he needs the sine function, he must develop the series.

By involving the student completely in individual research, in a problem-solving situation, he must make his own critical decisions. Which attack on a problem will be most fruitful? What steps must be taken in which order? What are personal weaknesses which must be overcome to succeed? The fact that a powerful computer can obediently do his will, if properly addressed, has a motivational quality much like that teenagers feel for automobiles.

To obtain the six objectives set forth in Unit 1, problems and methods of presentation must be developed that make the most effective use of the capabilities of the computer. Following are some of the problems and methods that have been used successfully in obtaining these objectives.

A. Skills

Example 1. The teacher programs multiplication facts; for example 9 x 0 through 9 x 9. However, 9 x 4 is programed to give an answer of 35. The Wyle card is programed to stop when the multiplication problem appears on the display screen. The pupil then mechanically places his answer into R 1 (register #1). The card reader is then punched, and the programed answer appears in the accumulator. The pupil compares the answer in the accumulator with his answer in R 1. If the answers are the same, he continues to the next problem; if they are different, he challenges the machine. To do this, he must recognize that multiplication is repeated addition. In the problem of $9 \times 4 = 35$, the pupil must add $9+9+9+9$ in order to determine whether he or the machine is correct. He should do this addition with pencil and paper.

Example 2. The following homework assignment may be given: Make up any problem in an operation of arithmetic and check the answer on the computer when you come to class. Pupils will add a group of numbers containing 7 rows and 7 columns; they will multiply 7 digits by 7 digits; they will divide 10 digits by 5 digits, etc. This is true of pupils who would not ordinarily do a single homework problem no matter how easy it was. It is the challenge of the computer that gets them. No pupil wants to check out an easy problem, and all pupils look forward to checking the answer on the computer.

Example 3. Divide 2600 by 145 and give the answer as a mixed number. The answer to this problem is $17\frac{135}{145}$. To check this answer on the Wyle Scientific, the fraction must be changed to a decimal, since the Wyle does not deal in fractions. The amount of computation that is necessary in a problem of this type is obvious. Again, due to the motivation of the computer, pupils will work on this type of problem long and diligently.

B. Concept Formation

Example 1. Program the formula for perimeter of a rectangle in four different ways.

The formula would take the following forms which would be programed:

$$P = 2L + 2W$$
$$P = 2W + 2L$$
$$P = 2\ (W + L)$$
$$P = (W + L)\ 2$$

The properties of commutativity of addition and multiplication can be explored as well as factoring.

Most pupils will remember this formula, after programing it in the above four ways, without trying to "formally" commit it to memory.

Example 2. Program the formula for the area of a square for:

$$S = 2,\ 5,\ 9, \text{ and } 16$$

Substituting the values for S in the formula $A = S^2$ should begin to develop in the pupil an idea of variable. As we vary S, A varies. It may be desirable, at this point, to introduce the concepts of dependent and independent variable and show how the value of A depends on the value assigned to S. Therefore, A is called the dependent variable and S the independent variable.

Example 3. Give an addition problem involving 5 rows and 5 columns. Have the pupils find the sum. Now ask whether the sum would be the same if the addition were done from the bottom row going up. Let the pupils check this out with pencil and paper. When they are convinced that the sum is the same, interchange the places of the first two numbers and add. Allow the pupils to use the computer now and check the sum for all possible combinations of these five addends. The "add any register" can be used very effectively in this type of problem.

Example 4. Place 534 into Ent; 247 into Acc; 700 into R1; 70 into R2; 11 into R3. These two sums should be the same. Why?

A variation of the above problem would be to place a different integer into R1, R2, and R3. The problem would be to find what integers must be placed in Ent and Acc so that R1 + R2 + R3 would be equal to Ent plus Acc.

Example 5. Dividing 2600 by 145 gives a quotient of 17 and a remainder of 135. Check this answer on the computer.

Place 17 into M Q
145 into Ent
135 into Acc
Depress "Multiply and Add" button
2600 will appear in the Acc

When the pupils can perform the above five steps, a clear understanding of division and remainders is shown.

C. Problem Solving

This is, without a doubt, the most difficult aspect of mathematics to teach. The computer offers an opportunity for a fresh and novel approach to the solution of word problems. Through the use of flow charting, as described in IV of Unit 1, the problem can be changed from one of words to one of simply identifying the operation needed for the solution. Through this approach, pupils with reading difficulties or under-achievers may advance to the solution of more difficult types of problems.

The following type of problem, which can be used with the more capable pupils, is more readily understood when a computer is used as an aid in the solution. These problems are classed as Diophantine-type problems. They are easily solved on a computer with a cut-and-try approach. However, as will be seen, understanding and insight must be developed before the cut-and-try is attempted. An example of this type of problem and its solution follows.

Example 1. A farmer has $100 and wishes to buy exactly 100 farm animals. Chickens cost 50 cents each, horses $10 each, and turkeys $3 each. He must buy at least one of each. How many of each does he buy?

Let us first solve this problem through the use of algebra and then see how we could have solved it, in a much simpler manner, using the computer.

Let $c =$ number of chickens
$h =$ number of horses
$t =$ number of turkeys

Then, $c + h + t = 100$
$.5c + 10h + 3t = 100$

Eliminating c in these equations we obtain
$5t = 100 - 19h$

We now have one equation in two unknowns, which seems to have an infinite number of solutions. However, there is another condition on t and h which aids in the solution. This condition is that t and h are both positive integers. Now, since there are two conditions and two unknowns, we know there may exist a unique solution to this equation.

Solve the equation for t

$$t = \frac{100 - 19h}{5}$$

Since t must be integral, the right member of the equation must also be integral. We can then separate the right member into two parts; one part already integral, the other to be made integral:

$$t = \frac{100}{5} - 3h - \frac{4}{5}h$$

$\frac{100}{5}$ and 3h are integral, since h is. Therefore, $\frac{4}{5}h$ must be made integral by choosing a suitable value of h.

$$\text{Let } \frac{4}{5}h = I$$

$$h = \frac{5}{4}I = I + \frac{1}{4}I$$

Continuing in the above manner:

$$\frac{1}{4}I = M$$

$$I = 4M$$

The smallest integer for M that would make I an integer is 1. Therefore, set $M = 1$. Substituting values back into the various equations, we obtain the answer to our problem, which is: $h = 5$
$t = 1$
$c = 94$

We could have used the equation $5t = 100 - 19h$ and solved this problem on the computer. Knowing that t and h must be integral, and also recognizing that h cannot be larger than 9, it is a simple problem to obtain the answer on the computer through the cut-and-try method.

Another type of problem, with which the computer can be used effectively, follows.

Example 2. The integers, 1, 3, 8, N have the property that the product of any two when added to unity yields a square. What is N?

This problem asks that the three following conditions be satisfied:

1. $N + 1 =$ perfect square
2. $3N + 1 =$ perfect square
3. $8N + 1 =$ perfect square

Now N could equal any of the following numbers.
N = 15, 24, 35, 48, 63, 80, 90, 120, 123, 165, etc.

However N cannot be any integer not listed above.
The reason for this is as follows:

$$\begin{aligned} 15 + 1 &= 16 = 4^2 \\ 24 + 1 &= 25 = 5^2 \\ 35 + 1 &= 36 = 6^2 \\ &\text{etc.} \end{aligned}$$

Performing the necessary operations on the computer, we find that 120 is the integer which satisfies the three conditions given above.

D. Individual Exploration and Research

Example 1. Using the computer, multiply 143 by 1 through 9 and then multiply each product by 7. For example:

$143 \times 1 = 143$ $143 \times 7 = 1001$
$143 \times 2 = 286$ $286 \times 7 = 2002$

The pattern should be recognized before multiplication by 9 is reached.

Example 2. Using the computer, find the pattern in the following:

$$\begin{aligned} 7 \times 7 &= 49 \\ 67 \times 67 &= 4489 \\ 667 \times 667 &= 444889 \end{aligned}$$

Example 3. Using the computer, find the pattern for the following:

$$\frac{22 \times 22}{1 + 2 + 1}$$

$$\frac{333 \times 333}{1 + 2 + 3 + 2 + 1}$$

$$\frac{4444 \times 4444}{1 + 2 + 3 + 4 + 3 + 2 + 1}$$

$$\frac{999999999 \times 999999999}{1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1}$$

Example 4. Find the formula for the sum of an arithmetic progression:

Put the integer 1 into the Acc. and then add 2, 3, 4, etc. List the partial sums as follows:

$1 = 1$

$1+2=3$
$1+2+3=6$
$1+2+3+4=10$

$$1+2+3 \;\ldots\ldots\; n = n\,\frac{(n+1)}{2}$$

In like manner, the formula for a geometric progression could be found.

E. Develop Desirable Habits

Flow charting and writing programing instructions on the "Wyle Coding Sheet" furnish practice in organizing one's thoughts and in putting them down in a neat, logical order.

Programing on the Wyle program card takes care and accuracy.

Programing is difficult, so a pupil soon learns to be very careful in his work and strives to make as few errors as possible. Programers rarely obtain a correct program the first time through. Debugging takes place following or during a computer run. It is this attitude of working with the computer to achieve results that may be considered positive. Debugging is a necessary check in trying to write a program. It is perfectly normal to make mistakes; that is precisely the reason for debugging.

The loss of a homework paper or of a program card could mean that the pupil would not be permitted to use the computer that day. Consequently, pupils very rarely come to class without the homework assignment and almost never lose the program card.

F. Enhance Self-Image

All that is necessary for enhancing that self-image is for the pupil to be capable of writing a program and seeing it work on the computer. The Wyle Scientific uses a very simple language and consequently is very easy to program. Every pupil should meet with success.

PROBLEMS

1. Divide 3700 by 133. Give the answer as a mixed number and check this answer on the computer.

2. The area of a triangle is $\frac{1}{2}$ bh where b is the base and h is the altitude. Write a program for this formula in four different ways.

3. Write a program for the average of n numbers.

4. Write programs and flow charts for the following formulas.

   A. $S = P\,(1 + i)^{w}$

   B. $M = \sqrt{T/75}$

   C. $a = \sqrt{b^2 + c^2 - 2\,bc\cos A}$ (Law of Cosines)

UNIT 5

NUMBER NOTATION SYSTEMS

Part I. Decimal Number System

To understand the internal operation of many digital computers, it is essential to understand scales of notation other than the decimal scale. The most widely used scales at the present time are binary, octal, and hexadecimal. Since all scales of notation are directly related and since we are most familiar with the decimal system, we will just relate these scales to the decimal scale and then to each other.

Any system of notation is essentially a means of counting. For example, in the decimal system we use 10 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Notice, in Figure 5-1 (Page 56), there is no repetition of any digit in the B^0 column in the first ten digits. Then the digits are repeated. In column B^1 each digit is used 10 times before the next one is used 10 times. In column B^2 each digit is used 100 times before the next digit is used 100 times. In the B^3 column each digit is used 1000 times, etc.

Using Figure 5-1, we can represent any number. For example, 345 can be represented as

$$\begin{array}{ccc} B^2 & B^1 & B^0 \\ \hline 3 & 4 & 5 \end{array}$$

In other words, we take the 5 from B^0, the 4 from B^1, and the 3 from B^2. In this case, the base (B) is 10, so this number can be represented as follows:

$$3\ (10)^2 + 4\ (10)^1 + 5\ (10)^0$$

The general representation of any integer I can be written as follows:

$$I = C_n\ (10)^n + \cdots + C_2\ (10)^2 + C_1\ (10)^1 + C_o\ (10)$$

Part II. Binary and Octal Number Systems

Figure 5-2 (Page 57), shows a comparison of the Binary and Octal number systems and their relationship to the decimal system.

Referring back to Figure 5-1, we recall that when $B = 10$, the B^0 column repeats the ten digits (0 to 9) after the first ten digits, and it repeats these digits after each succeeding ten digits. Notice also that the digits in this column change every B^0 times, or every one time. The digits in the B^1 column change after being used 10 times, etc. Now, referring to Figure 5-2, it is readily seen that for $B = 8$, the B^0 or 8^0 column repeats the first 8 digits (0 to 7) after the first 8 digits, and it repeats these digits after each succeeding eight digits. The digits in this column change every 8^0 times, or every one time. The digits in the $B^1 = 8^1$ column change after being used 8 times, the digits in the $B^2 = 8^2$ column change after being used 8^2, or 64 times, etc. In the binary table, it is readily seen that this same pattern exists. The $B^0 = 2^0$ column changes every one time; in the $B^1 = 2^1$ column, the digits change after being used two times; in the $B^2 = 2^2$ column, the digits change after being used four times; in the $B^3 = 2^3$ column, the digits change after being used 2^3, or 8 times; etc.

The general representation of a number in any base is:

$$\cdots\cdot + N_3\ B^2 + N_2\ B^1 + N_1\ B^0 + M_1B^{-1} + M_2\ B^{-2} + \cdots\cdots$$

where B represents the base being used, N_1 represents the units digits, N_2 the tens digit, N_3 the hundreds digit, etc.; and M_1 represents the tenths digit, M_2 the hundredths digit, etc.

There is a very interesting and useful relationship between binary and octal numbers, as shown in Figure 5-2. This relationship shows how certain concepts can be tied together and expanded upon to produce a result which at first glance seems an astounding coincidence, but which, upon further study, is readily accepted as a natural outgrowth of certain mathematical laws. Therein lies the beauty, grace, mystery, and majesty of mathematics.

In order to investigate this relationship, it is necessary to introduce the terms "bit" and "bit pattern" into our mathematical vocabulary.

A "bit" is a binary digit (0 or 1).
A "bit pattern" is a combination of N binary digits.

For example: 00 is a 2-bit pattern.
000 is a 3-bit pattern.

Any "bit pattern" of N binary digits represents 2^N possible choices; e.g., a 2-bit pattern represents 2^2, or 4 possible combinations, as follows:

00
01
10
11

A 3-bit pattern represents 2^3, or 8 possible combinations, as follows:

000
001
010
011
100
101
110
111

Now, suppose we list the binary numbers shown in Figure 5-2, in a 3-bit pattern, and the equivalent octal numbers next to them, as in Figure 5-3 (Page 57). Notice that we have one octal digit for each 3-bit pattern in the equivalent binary number. Furthermore (and this is the interesting and important part), each 3-bit pattern of a binary number represents the value of the pattern, in binary, as an octal digit. This permits us to interchange binary and octal numbers by sight.

For example, the binary number 010000000011 may be displayed in 3-bit pattern as:

010 000 000 011 .
2 0 0 3

Under each "bit pattern" is the value of that 3-bit pattern taken by itself as a binary number. The numeral 2003 which was obtained in this manner is the octal representation of 010000000011 (base 2).

Part III. Hexadecimal (base 16) Number System

If we desire to interchange binary and hexadecimal numbers by sight, the 2^N becomes 2^4, since the hexadecimal number system has 16 digits and 2^4 is equal to 16. Therefore, $N = 4$, which means we will use a 4-bit pattern in our binary numbers. Figure 5-4 (Page 59) uses this 4-bit pattern.

Notice that all "bits" have been used; we have a total of 2^4, or 16, different "bits." Notice also that we have skipped the number 10 and used A for the hexadecimal equivalent of 1010 in base 2. The reason for this becomes clear if we look at the binary number 10000 in Figure 5-4. 10000 in base 2 is equivalent to 16 in base 10, and we desire it to be 10 in base 16.

Part IV. Comparing Number Systems

The teacher must not lose sight of his primary purpose: mathematics education. While binary, octal, and hexadecimal number systems can be interesting, the teacher must continue to draw parallels between these systems and the decimal system. Studying other systems highlights the meaning of place value. It can also be used to clarify the multiplication and division algorithms we use.

DECIMAL SYSTEM OF NUMERATION

B^3	B^2	B^1	B^0
	0	0	0
	0	0	1
	0	0	2
	0	0	3
	0	0	4
	0	0	5
	0	0	6
	0	0	7
	0	0	8
	0	0	9
	0	1	0
	0	1	1
	0	1	2
	0	1	3
	0	1	4
	0	1	5
	0	1	6
	0	1	7
	0	1	8
	0	1	9
	0	2	0
	0	2	1
	0	2	2
	0	2	3
	0	2	4
	0	2	5
	0	2	6
	0	2	7
	0	2	8
	0	2	9
	0	3	0
	0	3	1
	0	3	2
	0	3	3
	0	3	4
	0	3	5
	0	3	6
	0	3	7
	0	3	8
	0	3	9
	0	4	0
	0	4	1

Figure 5-1

COMPARISON OF DECIMAL, OCTAL, AND BINARY SYSTEMS OF NUMERATION

Decimal	Octal			Binary					
	8^2	8^1	8^0	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0						0
1	0	0	1						1
2	0	0	2					1	0
3	0	0	3					1	1
4	0	0	4				1	0	0
5	0	0	5				1	0	1
6	0	0	6				1	1	0
7	0	0	7				1	1	1
8	0	1	0			1	0	0	0
9	0	1	1			1	0	0	1
10	0	1	2			1	0	1	0
11	0	1	3			1	0	1	1
12	0	1	4			1	1	0	0
13	0	1	5			1	1	0	1
14	0	1	6			1	1	1	0
15	0	1	7			1	1	1	1
16	0	2	0		1	0	0	0	0
17	0	2	1		1	0	0	0	1
18	0	2	2		1	0	0	1	0
19	0	2	3		1	0	0	1	1
20	0	2	4		1	0	1	0	0
21	0	2	5		1	0	1	0	1
22	0	2	6		1	0	1	1	0
23	0	2	7		1	0	1	1	1
24	0	3	0		1	1	0	0	0
25	0	3	1		1	1	0	0	1
26	0	3	2		1	1	0	1	0
27	0	3	3		1	1	0	1	1
28	0	3	4		1	1	1	0	0
29	0	3	5		1	1	1	0	1
30	0	3	6		1	1	1	1	0
31	0	3	7		1	1	1	1	1
32	0	4	0	1	0	0	0	0	0
33	0	4	1	1	0	0	0	0	1
34	0	4	2	1	0	0	0	1	0
35	0	4	3	1	0	0	0	1	1
36	0	4	4	1	0	0	1	0	0
37	0	4	5	1	0	0	1	0	1
38	0	4	6	1	0	0	1	1	0
39	0	4	7	1	0	0	1	1	1
40	0	5	0	1	0	1	0	0	0
41	0	5	1	1	0	1	0	0	1

Figure 5-2

COMPARISON OF BINARY AND OCTAL NUMBERS

Binary number	Octal number
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7
001 000	10
001 001	11
001 010	12
001 011	13
001 100	14
001 101	15
001 110	16
001 111	17
010 000	20

Figure 5-3

COMPARISON OF BINARY AND HEXADECIMAL NUMBERS

Binary	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F
0001 0000	10
0001 0001	11
0001 0010	12
0001 0011	13
0001 0100	14
0001 0101	15

Figure 5-4

UNIT 6

AIDS TO USING A COMPUTER

Part I. Subroutines

Suppose we had a code in which the same set of instructions was needed in two or more places. These instructions could be copied into each place, but this would be a waste of storage space and of our own time. Also, it would invite error as we repeated the same set of instructions. In our discussion on "looping," we showed that whenever we repeat the same job, a better method must exist. One of these better methods is subroutining, which permits us to write the instructions once, rather than each time they are needed.

Subroutine subprograms can be completely independent of the main program; yet, it is easy to set up "Communication" between the main program and the subprograms. This means that a large program can be divided into parts that can be compiled individually, making it possible to correct errors in one subprogram without recompiling the entire program.

Subroutines are presently in such demand and their use so widespread that libraries are being developed where certain subroutines are available.

To use a set of instructions as a subroutine, we must always do two things:

1. Arrange to return to the mainstream of code immediately following the subroutine instructions.
2. After the desired instructions are transmitted to the subroutine, the result that is obtained is transmitted back to the mainstream of code.

There are two types of subroutines:

1. Open subroutine--This is a subroutine that is inserted into the main program at a point where it is used. This type is used when the subroutine is used only once.
2. Closed subroutine--This is a subroutine that is used as often as is necessary within one main program.

Example:

Compute $Z = C \text{ Cos } \emptyset + P^n R \text{ Cos } 4$, where C, N, and R are constants, and $\emptyset$, P, and 4 are variables. Use N log P for P^n.

Note that we have to compute the cosine twice; once for Cos $\emptyset$, and again for Cos 4. We resort to subroutining: The subroutine can take the form of open or closed; in each case it would look as shown in Figures 6-1 and 6-2.

Open Subroutine

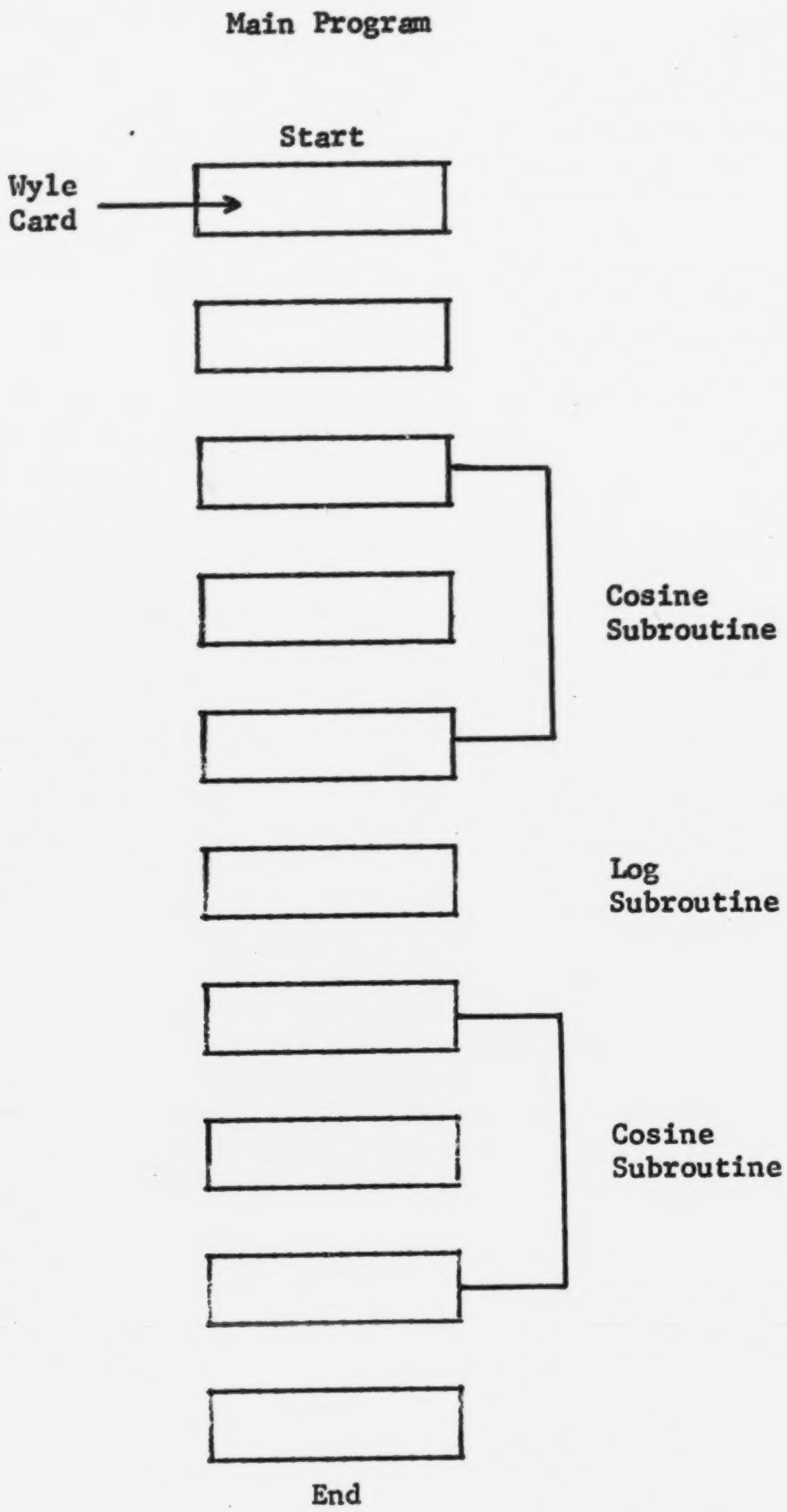


Figure 6-1

Closed Subroutine

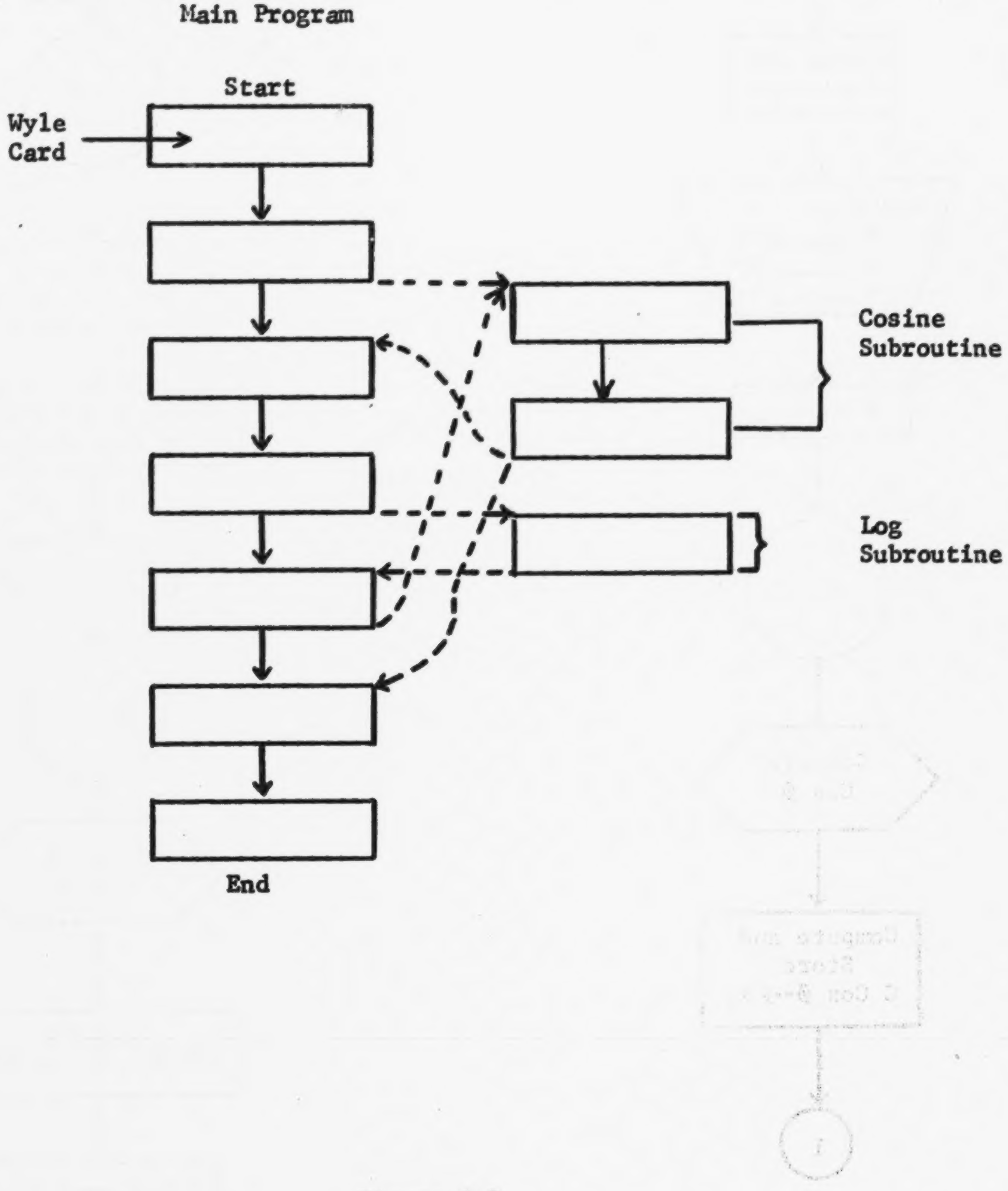


Figure 6-2

The flow chart for programing this equation appears as follows:

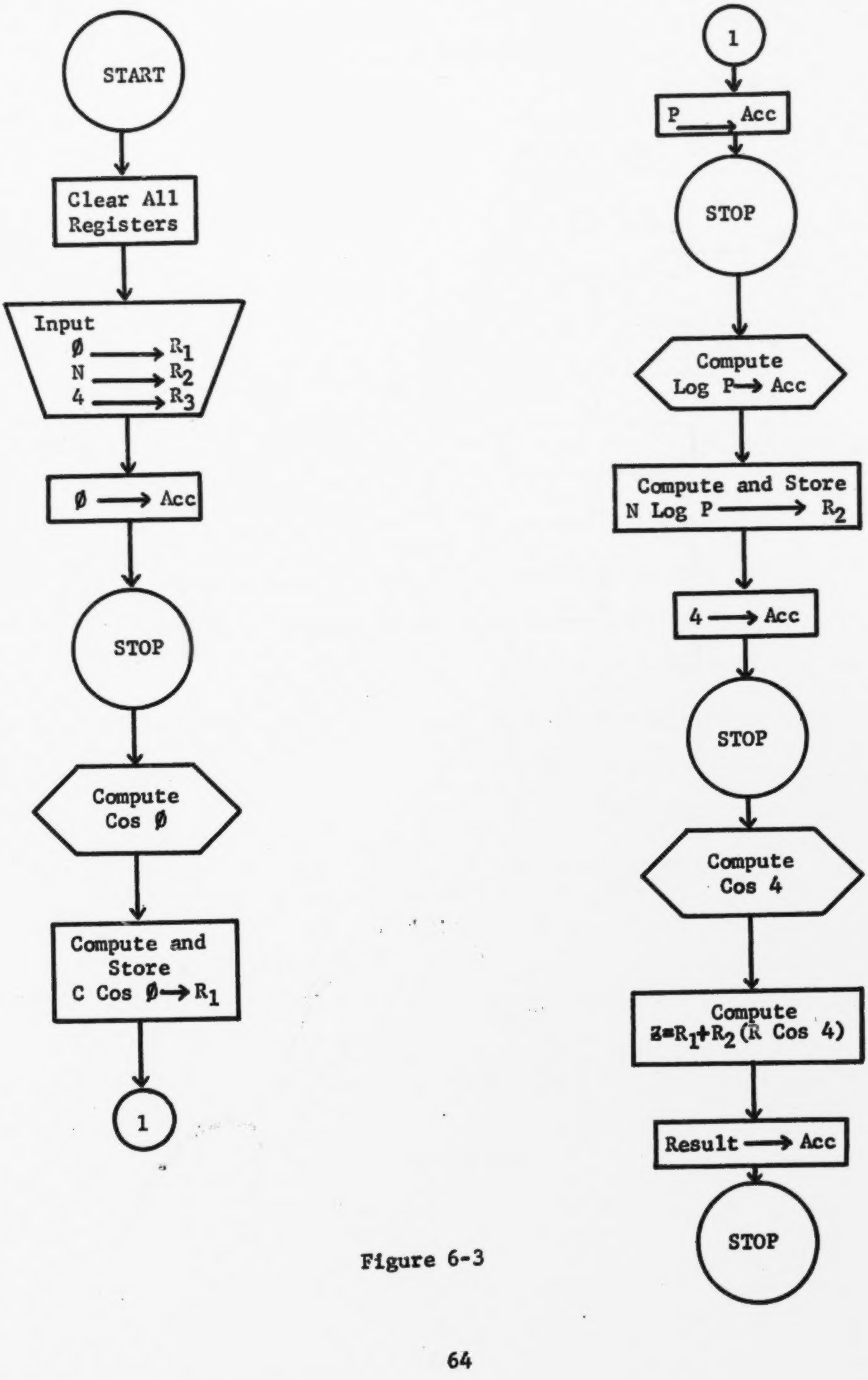


Figure 6-3

Part II. Transparencies

Transparencies can be used very effectively in conjunction with a computer in a mathematics classroom. Copies showing the content of the transparencies that have been developed for this program are included in this unit. However, teachers, as they become familiar with this program and recognize its potential, should develop additional transparencies.

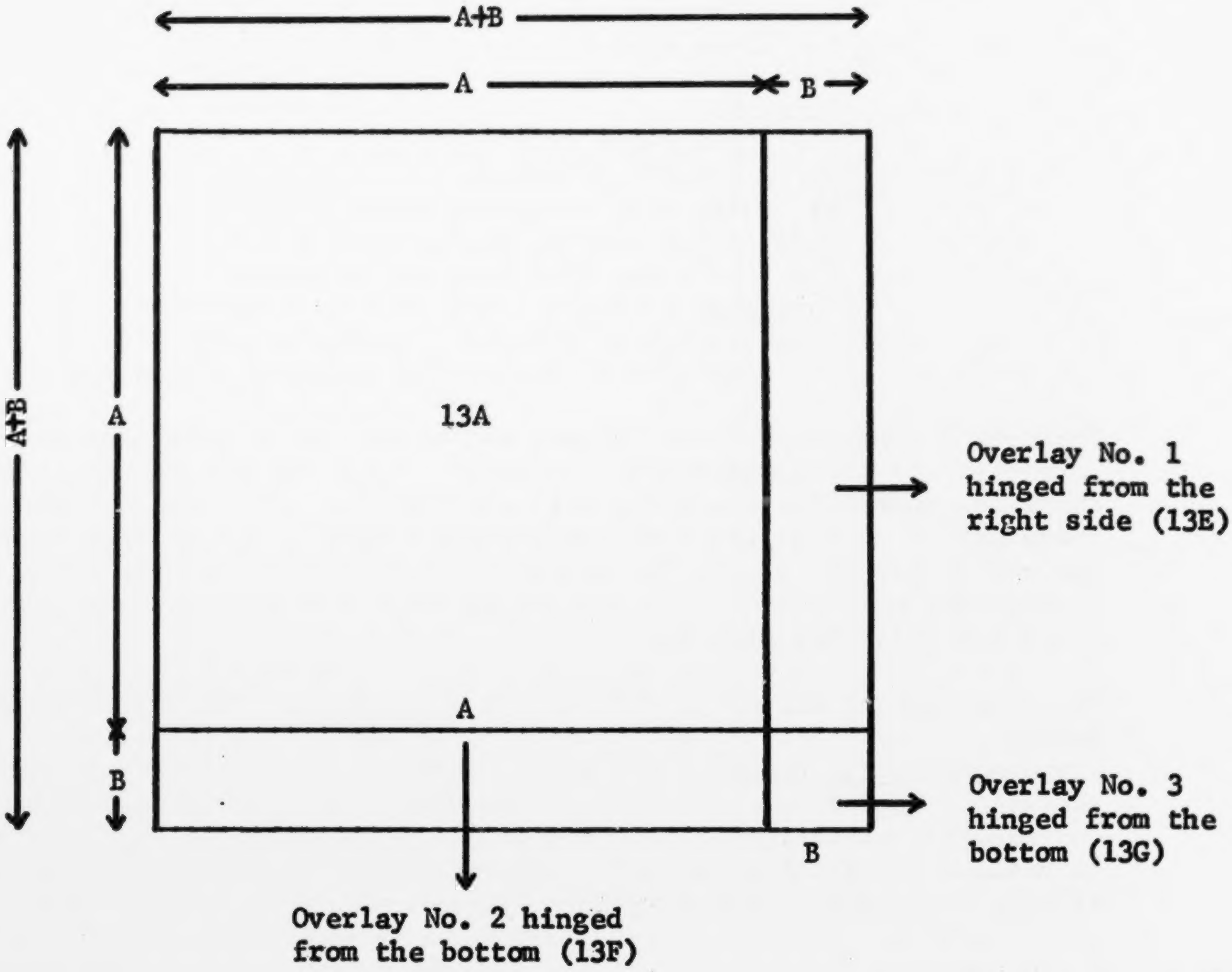
The following is a list of the materials necessary for producing transparencies.

- Mounting frames
- Instructo color transparent pencils
- Pen, wick, disposable, for overhead transparencies (5 colors) - chisel point and fineline
- Cellulose acetate, clear, .005" (10"x10")
- Tape, page mending clear, colorless 3/4" wide, Scotch #810.
- Scissors
- 1/4" Grid paper (for masters)
- No. 2 pencils - Scripto
- 12" ruler with lettering guide
- Film, transparency, thermofax type 127
- Trans - Hinges (for hinging overlays)
- Lettering pens and ink (Leroy or Koh-I-Noor)
- Film, adhesive, colored. Thermofax type 720. (4 colors, for adding color to transparencies).

Fourteen transparencies are recommended as an aid in using the Wyle Scientific in the mathematics classroom. Copies of the masters for these transparencies accompany this publication. The teacher may use these copies to make the fourteen transparencies. The masters numbered 1 through 9, 11, and 12 are for use specifically with the Wyle Scientific. The masters numbered 10, 13, and 14, however, may be used in a mathematics class not using any computer.

Should a teacher desire to eliminate the numbers at the bottom of the masters, he can easily block these numbers out with white tape. Transparencies 1, 2, and 3 are used to explain the physical makeup of the Wyle computer and to introduce the operations and machine language that is characteristic of this computer. Transparency 4 may be used to explain the function of the three arithmetic registers and the three storage registers. This transparency also should be used for pupil participation. Transparencies 5 and 6 are an extension of Transparency 4 and permit a greater variety of problems to aid the pupil in understanding the operations of the machine. Transparencies 7 and 8 are copies of the Wyle program card and are invaluable in explaining the function of all columns on this card. Transparency 9 is used as the coding sheet. Its use is shown in Transparency 11. However, we want a blank coding sheet transparency to use for pupil participation. It is advisable to run off a few hundred of these coding sheets, on a ditto machine, for pupil use in solving problems. Transparencies 10, 11, and 12 show the flow chart, coding, and program, respectively, of a typical expression. Transparency 13 consists of seven masters. However, the finished product of the seven masters consists of one transparency with six overlays. This transparency is used to show the visual representation of $(A+B)^2 = A^2+2AB+B^2$ and $(A-B)^2 = A^2-2AB+B^2$.

The visual representation of $(A+B)^2$ consists of a transparency with three overlays, as follows:



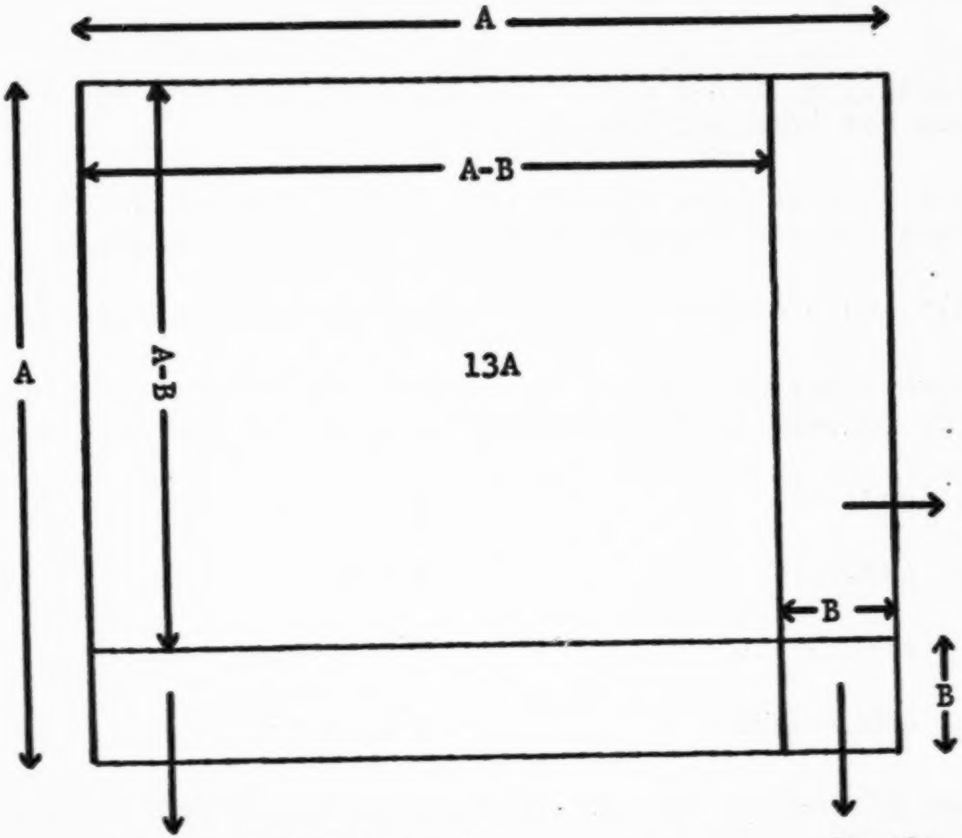
Overlay No. 1 hinged from the right side (13E)

Overlay No. 3 hinged from the bottom (13G)

Overlay No. 2 hinged from the bottom (13F)

Figure 6-4

The visual representation of $(A-B)^2$ consists of a transparency with three overlays, as follows:



Overlay No. 1 hinged from the left side (13C)

Overlay No. 2 hinged from the left side (13B)

Overlay No. 3 hinged from the left side (13D)

Figure 6-5

For easier viewing and better understanding, color should be used on Transparency 13. For example, 13A could be plain, 13E and 13R could be yellow, and 13G could be blue.

Transparency 14 is a visual representation of finding areas of regions by subtracting areas of subregions from the total region. In making this transparency, 14A is the static frame; 14B is hinged from the left side; and 14C is hinged from the right side.

Note that the masters for Transparencies 13 and 14 all have a small circle in the upper left and right corners. If these circles are matched on all overlays when they are hinged, the overlays will automatically fall into the proper position.

PROBLEMS

1. Make an addition and multiplication table in base 8.

2. Using the tables in Problem 1, make similar tables in base 2 and base 16 by sight.

3. Using Transparency 4, place 34.7 into MQ, 16.9 into ENT and 586.43 into ACC. What operation has been performed?

   Note that this problem should lead into a discussion of multiplication and division being inverse operations.

4. a. Program $(A+B)^2$ and $A^2+2AB+B^2$ separately and in the general form as given.

   b. Using the above programed cards, substitute the following values for A and B and find the answers to these expressions on the computer.

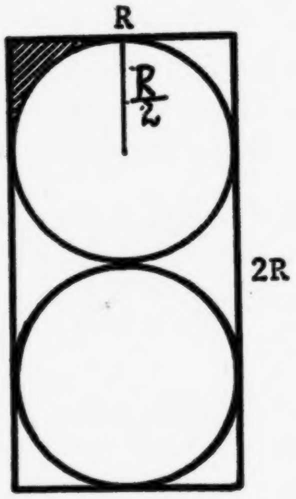
      1. $A=3$ $B=5$
      2. $A=3.9$ $B=5.6$
      3. $A=(4.3\ 5.6)$ $B=(7.8\ 3.4)$
      4. $A=3.7456859$ $B=4.87563$

   Note that discussion concerning results should follow, and then Transparency 13 should be shown and discussed.

5. a. Using Transparency 14 with $R=15$ and $\pi=10$, find the area between the two circles.

   b. Program the general formula for finding the area between the two circles.

6. a. Using Transparency 14, find the area of the shaded region in the following diagram.



   b. Program the formula for finding the shaded region in 6.a. in two different ways.

UNIT 7

FORTRAN PROGRAMING

Part I. The Compiler

Problem solving procedure must be presented to a computer in a language it can understand. A computer's basic language consists of elementary instructions, such as add, subtract, multiply, divide, shift a number left or right, punch a card, etc. The translation of these simple instructions for the computer is called coding, or programing. This can be carried out by a person, or the computer may assist in the process by use of a compiler. The compiler, in essence, translates the mathematical language used by the programer into a machine language understood by the computer.

Part II. Fixed and Floating Points

A. Fixed Points
A fixed point number is just an integer. It may be 0, positive, or negative, and its limit is based upon the machine being used.

B. Floating Point
A floating point number is treated like a fraction (between 0.1 and 1.0) times a power of 10. This number may be an integer or may have a fractional part, as is discussed later in this unit.

The computer automatically lines up decimal points before adding, subtracting, etc. This is the reason for the term "floating point." The number 3 is a fixed point constant; 3.0 is a floating point constant. These two numbers are not interchangeable, since the computer stores and processes them in entirely different ways.

C. Examples

Fixed point numbers or constants
0, +9, 203, -1463

Floating point numbers or constants
12.78, 0.0, 8.0, 4., -300., -.003, +.000006

In representing floating point constants, the letter E is used for exponentiation. Therefore, the above constants would be represented in the following way:

12.78 ⟶ .1278E2 ⟶ 1.278E1 ⟶ 1287.E-2
4. ⟶ 4. ⟶ .4E1
-.003 ⟶ -.3E-2 ⟶ -.03E-1 ⟶ -3.E-3

Note that any number represented in the above three groups is interchangeable within that group and may be used in the given form.

The decimal point must always appear in a floating point number. It may appear at the beginning of a number, at the end of a number, or between digits of a number.

Part III. Variables

A variable is a symbol or name which refers to a place in memory where the number or value represented by that symbol or name is stored.

The name of any variable consists solely of one or more alphabetic or numerical characters, the first of which must be alphabetic (A through Z), and the last of which must not be the letter F, since functions end in F. A variable may be a fixed or floating point quantity. If it is a fixed point variable, the name must start with any one of the following letters: I, J, K, L. M, or N. A floating point variable begins with any letter other than these six letters used for a fixed point variable.

Examples of fixed point variables:

I, KJM, MAL, M432, I7N2J

Examples of floating point variables:

AKV, B42J, C, D05

The following would not be interpreted as variables of either type, for the reasons given.

3AK - (begins with a constant)
*D05 - (begins with an operation)
E-F - (contains an operation)
G4.6 - (contains a decimal point)

The compiler places no significance on names; it merely inspects the first letter to see whether the variable is a fixed point or a floating point.

Part IV. FORTRAN Symbols

The FORTRAN symbols for the operations of arithmetic are:

Operation	FORTRAN Symbol
add	+
subtract	-
multiply	*
divide	/
exponentiation	**

The following are a few simple rules for writing expressions involving operations.

a. Parentheses are used in exactly the same manner as in mathematical notation: L*+B is invalid; L*(+B) is correct.

b. When it is not clear which operation comes first, the sequence is exponentiation, multiplication and division, addition and subtraction.

c. Among operations of equal precedence, the operations are performed from left to right. Example: X/Y*Z $= \frac{x}{y} \cdot z$

Although the above three rules are not all the rules involved in working with operations, they will suffice for our purposes.

Part V. Statements

DØ: The function of the DØ statement is control. With it we can control the repeated execution of succeeding statements in a manner ensuring useful results. It is a very powerful statement.

IF: This statement is the fundamental decision-making step of the FORTRAN language. Any question that is reducible to testing a numerical value for 0, negative or positive, can be expressed as an IF statement.

Example: IF (Z-A (J)) 10, 30, 20 is interpreted as follows: Whenever Z-A (J) $<$ 0 or, in other words, Z $<$ Aj, the next statement to be executed in sequence is the statement labeled 10. Whenever Z-A(J) is equal to zero execute the statement labeled 30. Whenever Z-A(J) $>$ 0 execute the statement labeled 20.

Note that A (J) means A sub j and is written Aj. K=A (J) means place the contents of location Aj into the memory location called K.

CØNTINUE: This statement means "Do nothing; just proceed from here." It is used to provide an executable statement with which to terminate a DØ.

INPUT: This statement specifies by name the items of information to be transferred from the input documents, e.g., punch cards. This statement also refers by number to a format code which describes the form of the input information. One of the commonly used forms of input is given by the statement READ.

Example: READ 11, A(J). This is an input statement ordering the computer to read in data. 11 refers to the format of the data and this data will be utilized whenever the program refers to A(J).

DIMENSION: This statement specifies how many items of data are to be used.

OUTPUT: This statement specifies the items that are to be retrieved from memory. A common form of output statement is PRINT.

Example: PRINT 21, Z. The format code is 21, and items are to be printed from memory location called Z.

The following problems and solutions may be an aid to better understanding of IV and V of this unit.

Example 1: Write the appropriate FORTRAN expression for the given mathematical expression. $C_i = \sqrt{(A_i)^2+(B_i)^2}$

Answer:
C(I)   (A(I)*A(I)+B(I)*B(I))**.5 or
C(I)   (A(I)**2+B(I)**2)**.5

Example 2: Find the largest number in a collection of 100 numbers. The flow chart for this problem follows:

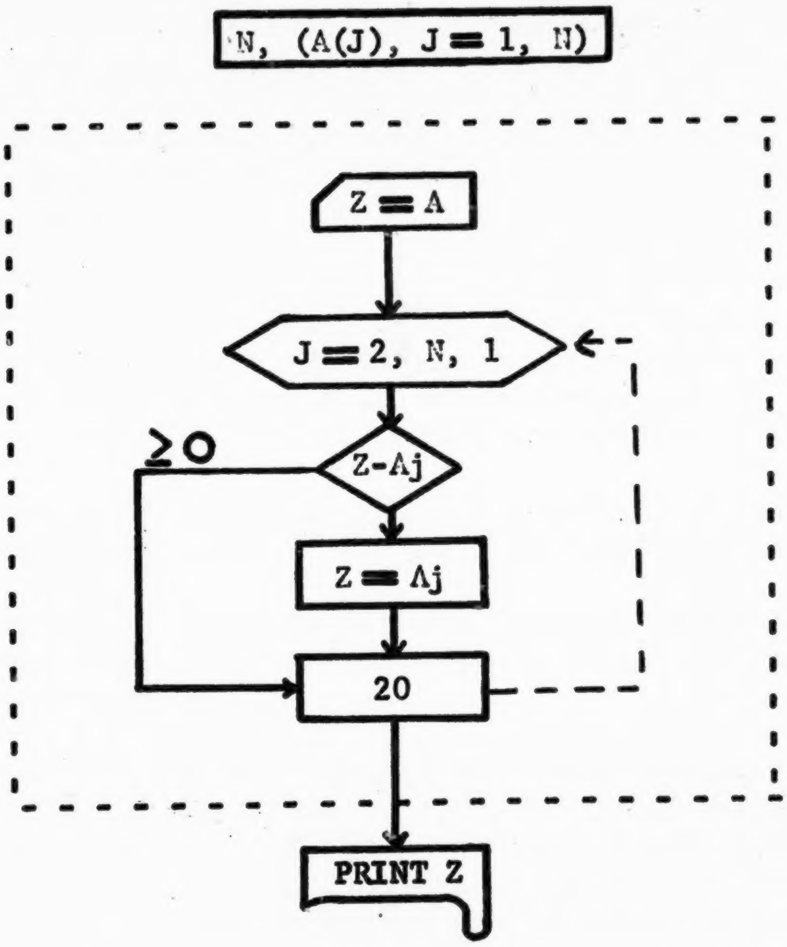


Figure 7-1

The FORTRAN language program for Example 2 is as follows:

```
     Label        Statement

                  DIMENSION A(100)
                  READ 11, A(J)
                  Z= A(1)
                  DØ20J  2, 100, 1
                  IF(Z-A(J) )10,20,20
       10         Z = A(J)
       20         CØNTINUE
                  PRINT 21,Z
```

The meanings of the above statements are as follows:

Line 1. There are 100 items.

Line 2. The format code is number 11, and we are inputting A(J).

Line 3. and Line 6. These two lines contain arithmetical substitution statements. Z = A(1) means place the contents of location called A(1) into memory location called Z. Z = A(J) means that the contents of A(J) are to be placed in Z. Therefore, Z has the current value of A(J).

Line 4. Execute repeatedly the statements which follow, down to and including the one labeled 20. Vary index J from an initial value of 2, in increments of 1, up through and including J = 100.

Line 5. Whenever $Z < Aj$, the next statement to be executed in sequence is 10. If $Z \geq Aj$, go to 20.

Line 7. Control passes to statement 20 either from the conditional IF statement, or from statement 10 immediately preceding 20.

Line 8. This is explained in OUTPUT statement in Section V.

The brief treatment given in this unit is not intended to give an understanding of the use of FORTRAN, but rather, is presented to give a glimpse of its range and usefulness.

APPENDIX

SOME POPULAR COMPUTER LANGUAGES

ALGOL	Algorithmic Language
BALGOL	Burroughs Corporation 220 Algol Compiler
COBOL	Common Business Oriented Language
FORTRAN	Formula Translation
JOVIAL	Jules (Swartz) Own Version of International Algebraic Language
MAD	Michigan Algorithm Decoder
NELIAC	Navy Electronics Laboratory International Algol Compilers
QUIKOMP	Designed by Monroe International to provide a rapid, simple means of using the Monrobot XI Computer as a mathematical tool
JOSS	Johnniac Open-Shop System

GLOSSARY

ACCUMULATOR REGISTER (ACC)	Used to hold the original number (radicand) in square root operations, the dividend in division operations, and the subtrahend in subtraction operations. The answers to all addition, subtraction, and multiplication operations of all types appear in this register.
ALGORITHM	A sequence of precisely defined steps showing the solution to a problem.
ANALOG COMPUTER	A computer which relies on some method of continuously measuring a physical quantity, such as weight, thickness, or resistance. Examples of this process are the odometer and the speedometer.
ARITHMETIC OPERATION	A computer operation using numerical calculations.
BINARY DIGIT	A digit in the binary scale of notation, such as zero or one.
BINARY NUMBER	A number written in binary notation bit; a binary digit.
CODE	The language used in preparing instructions for the computer.
COMPILER	A special program for a specific machine which translates instructions from some program language into the basic language of the specific machine.
COMPUTER	A device which accepts information, performs mathematical or logical operations with this information, and then supplies the results of these operations as new information.
DATA PROCESSING	A logical sequence of operations performed on data.
DIGITAL COMPUTER	A computer which represents information in discrete form. An example is the turnstile.

FIXED POINT	A fixed point number is just an integer. It may be zero, positive, or negative, and its limit is based upon the machine being used. Examples of fixed point numbers are 0, +9, 203, -1463.
FLOATING POINT	A floating point number is treated like a fraction (between 0.1 and 1.0) times a power of 10. This number may be an integer, or may have a fractional part. Examples of floating point numbers are 12.78, 0.0, 8.0, 4., -.003, +.00006.
FLOW DIAGRAM	A sequence of operations represented graphically.
HARDWARE	An array of processors and devices for storage or communication.
INITIALIZE	To set registers to their initial values immediately preceding a loop.
INPUT	Information fed into the internal storage of a computer.
LIBRARY	A collection of routines and subroutines which may be used in larger routines in a program.
LOOP	A coding technique whereby certain steps of a program are repeated. The sequence is then continued. (See Unit 3.)
MACHINE LANGUAGE	A language for programing instructions directly into a computer.
NUMERATION SYSTEM	A set of symbols used for naming numbers.
OUTPUT	Information transmitted from a computer to an output device.
OVERFLOW	This occurs when attempting an input of a quantity which is greater than the capacity of a register.

PROGRAM	A sequence of instructions for solving a problem; should include flow diagram.
READ	To transfer information from an input device to internal storage.
REGISTER	A device for storing information.
SOFTWARE	A collection of routines concerned with the system itself, rather than with applications.
STORAGE LOCATION	A place for storing data or instructions.
SUBROUTINE	A routine which can be made a part of a larger routine.
SYMBOLIC CODING	Instructions written in non-machine language.

TABLE OF POWERS OF 2

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

BIBLIOGRAPHY

Darnoivski, V. S. "Computers--Theory and Uses," National Science Teachers Association (1964)

Forsythe, A. "Mathematics and Computing in High School--A Betrothal," The Mathematics Teacher (January, 1964) pp. 2-7.

Goldstein, M. "Computer Languages," The American Mathematical Monthly, Vol. 72, No. 2, pp. 141-47.

Goldstine, H. H. "Calculating Machines," Encyclopaedia Britannica, Vol. 4., 1961 edition.

Gruenberger, Fred J. and Daniel D. McCracken, Introduction to Electronic Computers. New York: John Wiley and Sons, 1963. pp. 1-55 and pp. 123-33.

Hamming, R. W. "Impact of Computers," The American Mathematical Monthly, Vol. 72, No. 2, pp. 1-7.

Lovis, F. B. Computers 1. Boston: Houghton-Mifflin, 1964. 64 pp.

Lovis, F. B. Computers 2. Boston: Houghton-Mifflin, 1964. 70 pp.

Newman, James R. The World of Mathematics. New York: Simon and Schuster, 1956. Part XIX, Vol. 4, pp. 2066-2133.

Suppes, Patrick. "The Uses of Computers in Education," Scientific American (September, 1966). pp. 207-23.

Ulam, Stanislarv M. "Computers," Scientific American (September, 1964). pp. 203-16.

Weisert, C. H. "Computer Systems," The American Mathematical Monthly, Vol. 72, No. 2, pp. 150-56.

National Council of Teachers of Mathematics. Computer Oriented Mathematics. 1201 Sixteenth Street, N. W. Washington, D. C.: 1963. 182 pp.

TRANSPARENCY MASTERS

For Use With

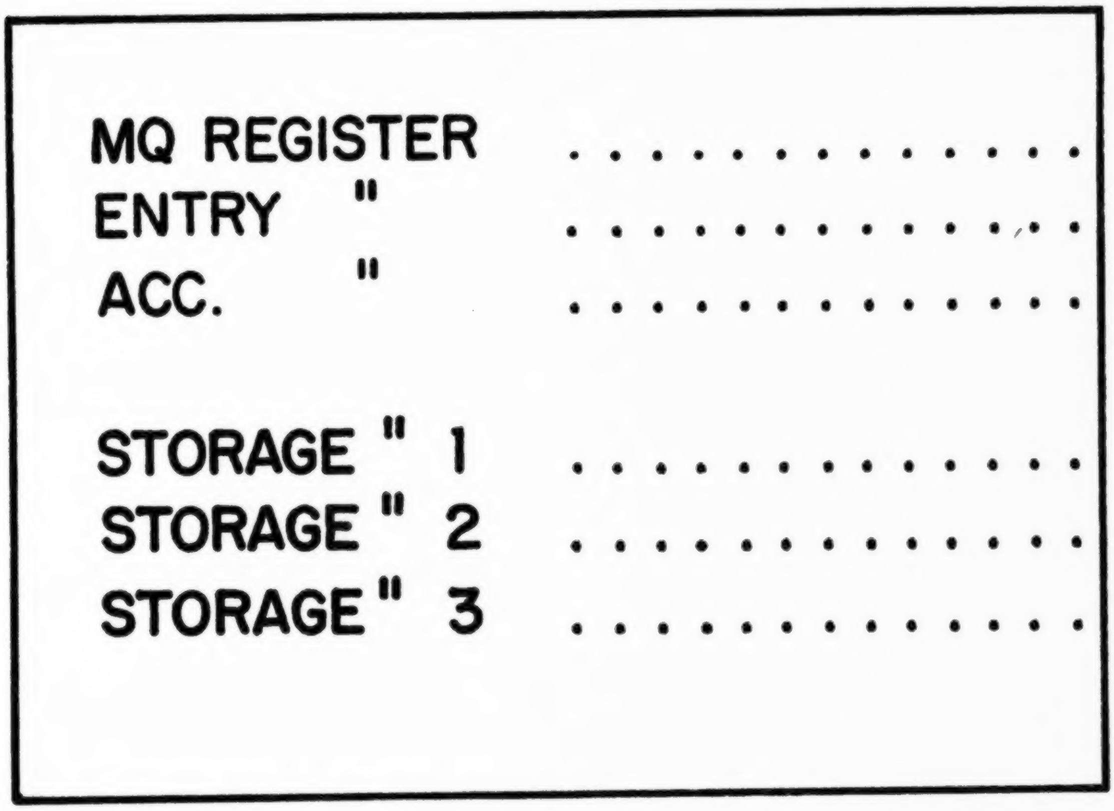
The Computer as an Aid in Teaching Mathematics

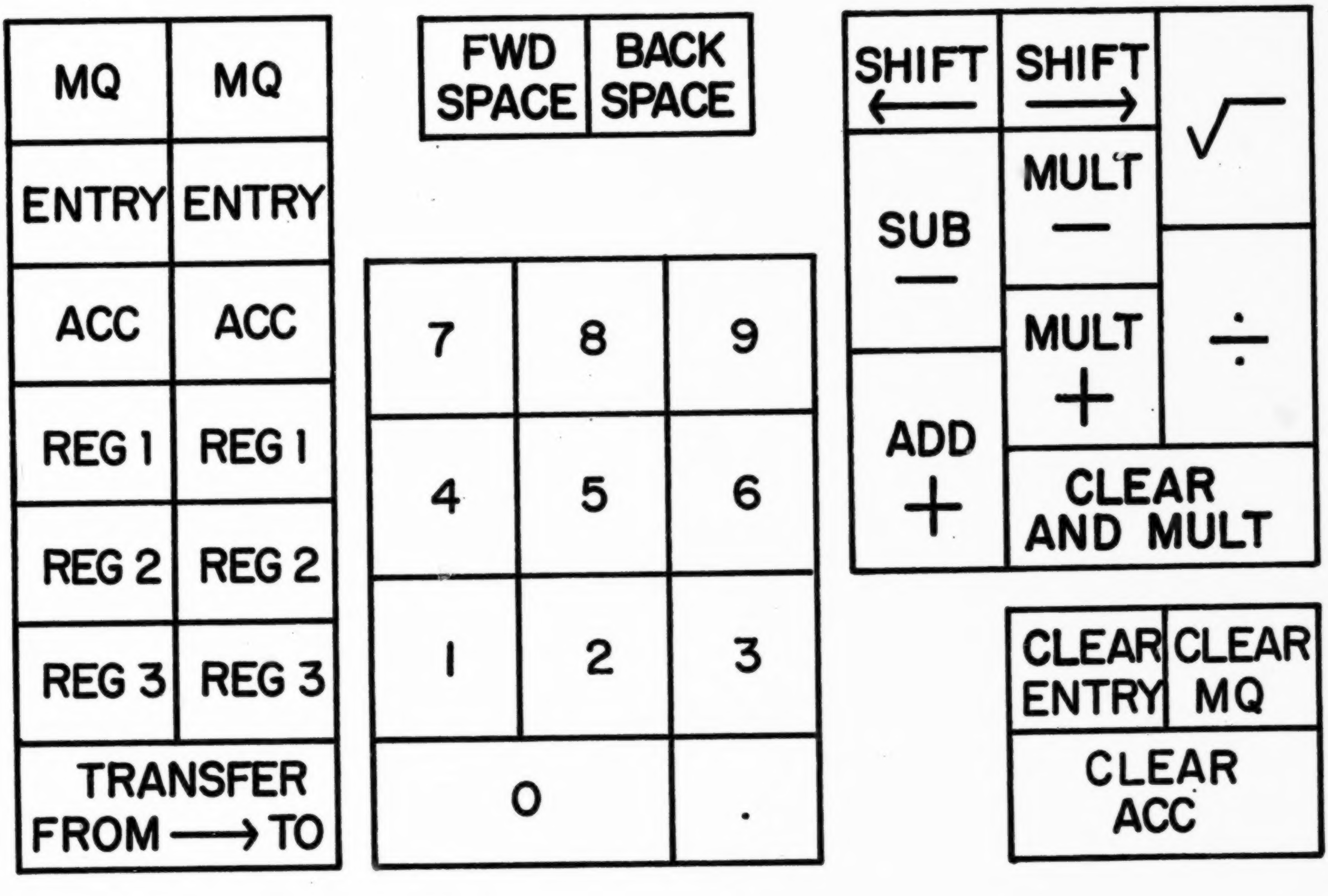
Instructional Bulletin

Grades 7 – 10

Division of Secondary Education
Specially Funded Programs
Publication No. ESEA 6
1967

[illegible] Schools


MQ REGISTER
ENTRY "
ACC. "
STORAGE " 1
STORAGE " 2
STORAGE " 3


MQ
MQ
ENTRY
ENTRY
ACC
ACC
REG 1
REG 1
REG 2
REG 2
REG 3
REG 3
TRANSFER FROM ⟶ TO
FWD SPACE
BACK SPACE
7
8
9
4
5
6
1
2
3
0
.
SHIFT ⟵
SHIFT ⟶
√
SUB −
MULT −
MULT +
÷
ADD +
CLEAR AND MULT
CLEAR ENTRY
CLEAR MQ
CLEAR ACC

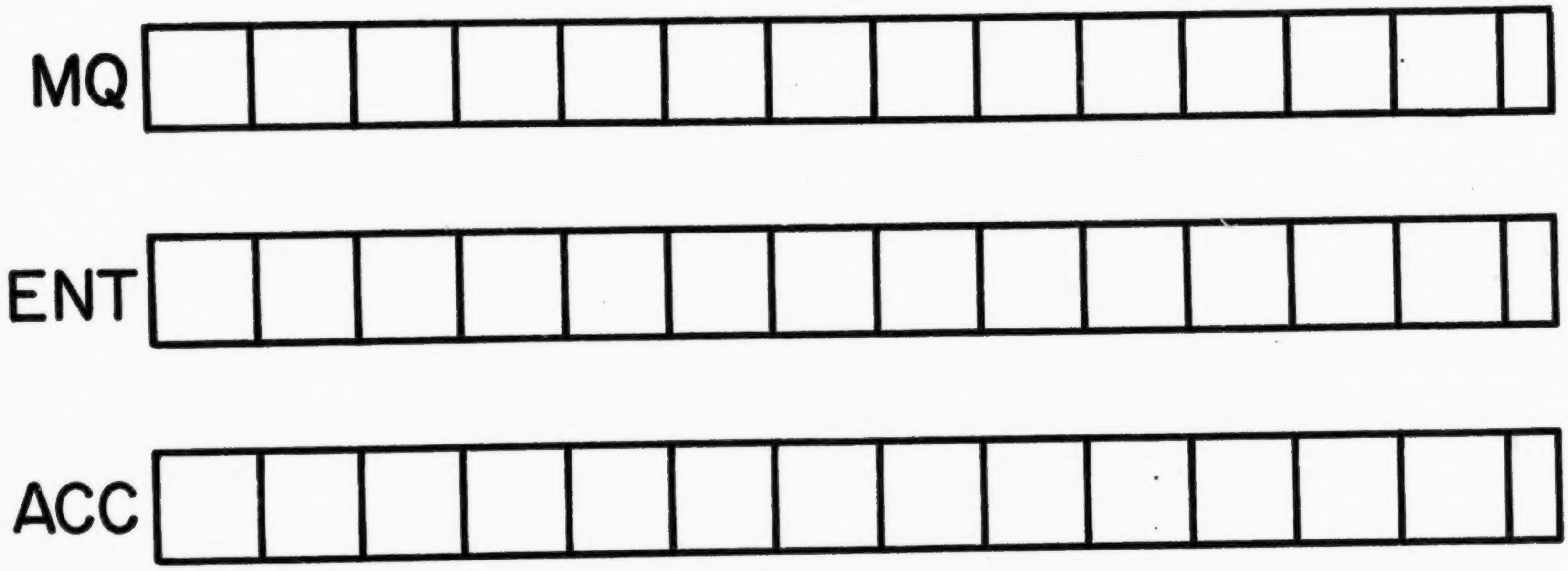
SUMMARY OF OPERATIONS

NORMAL OPERATING CONDITIONS

1. Decimal point in center (12th) position.
2. "Overflow Lock" Switch on (up).
3. "Add From Any Register" Switch (up).
4. "Keep Remainders" Switch off (down).

ARITHMETIC OPERATIONS

1. Addition $(\mathrm{ACC}) + (\mathrm{ENTRY}) \longrightarrow \mathrm{ACC}$
2. Subtraction $(\mathrm{ACC}) - (\mathrm{ENTRY}) \longrightarrow \mathrm{ACC}$
3. Clear and Multiply $(\mathrm{MQ}) \times (\mathrm{ENTRY}) \longrightarrow \mathrm{ACC}$
4. Multiply + $(\mathrm{ACC}) + \left[(\mathrm{MQ}) \times (\mathrm{ENTRY})\right] \longrightarrow \mathrm{ACC}$
5. Multiply − $(\mathrm{ACC}) - \left[(\mathrm{MQ}) \times (\mathrm{ENTRY})\right] \longrightarrow \mathrm{ACC}$
6. Divide $(\mathrm{ACC}) \div (\mathrm{ENTRY}) \longrightarrow \mathrm{MQ}$
7. Square Root $\sqrt{(\mathrm{ACC})} \longrightarrow \mathrm{MQ}$



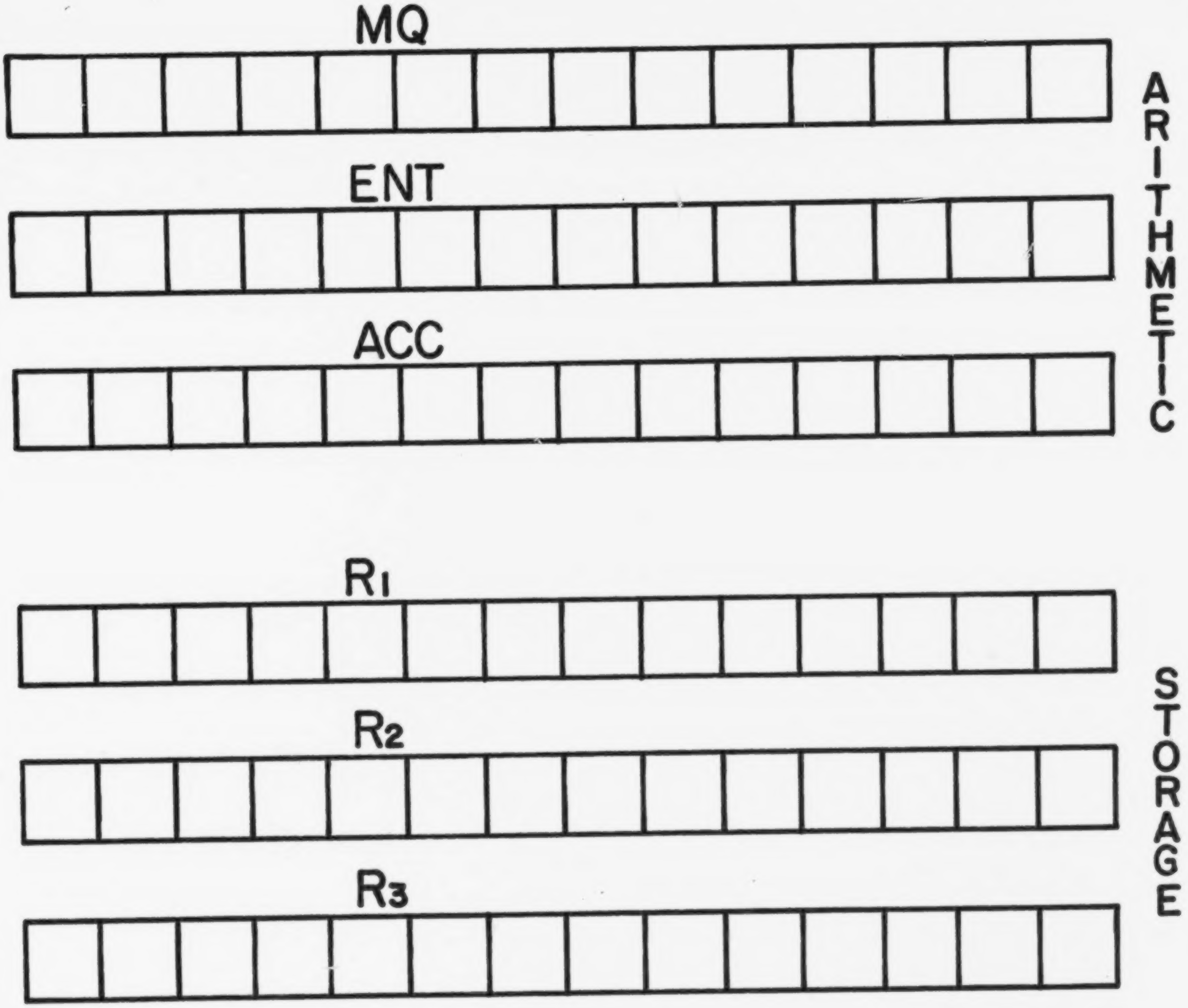
MACHINE OPERATIONS

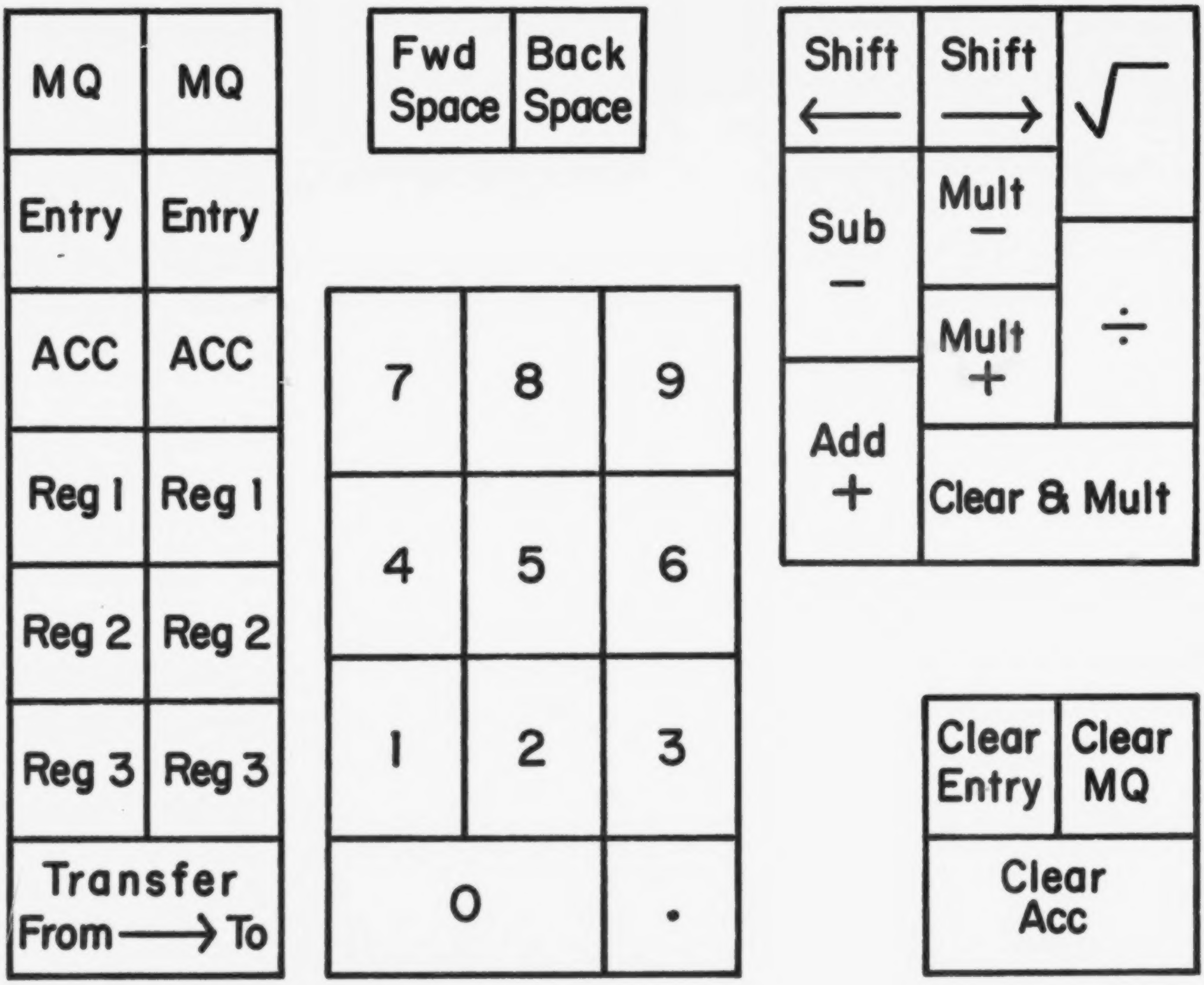
Addition: ACC + ENT → ACC

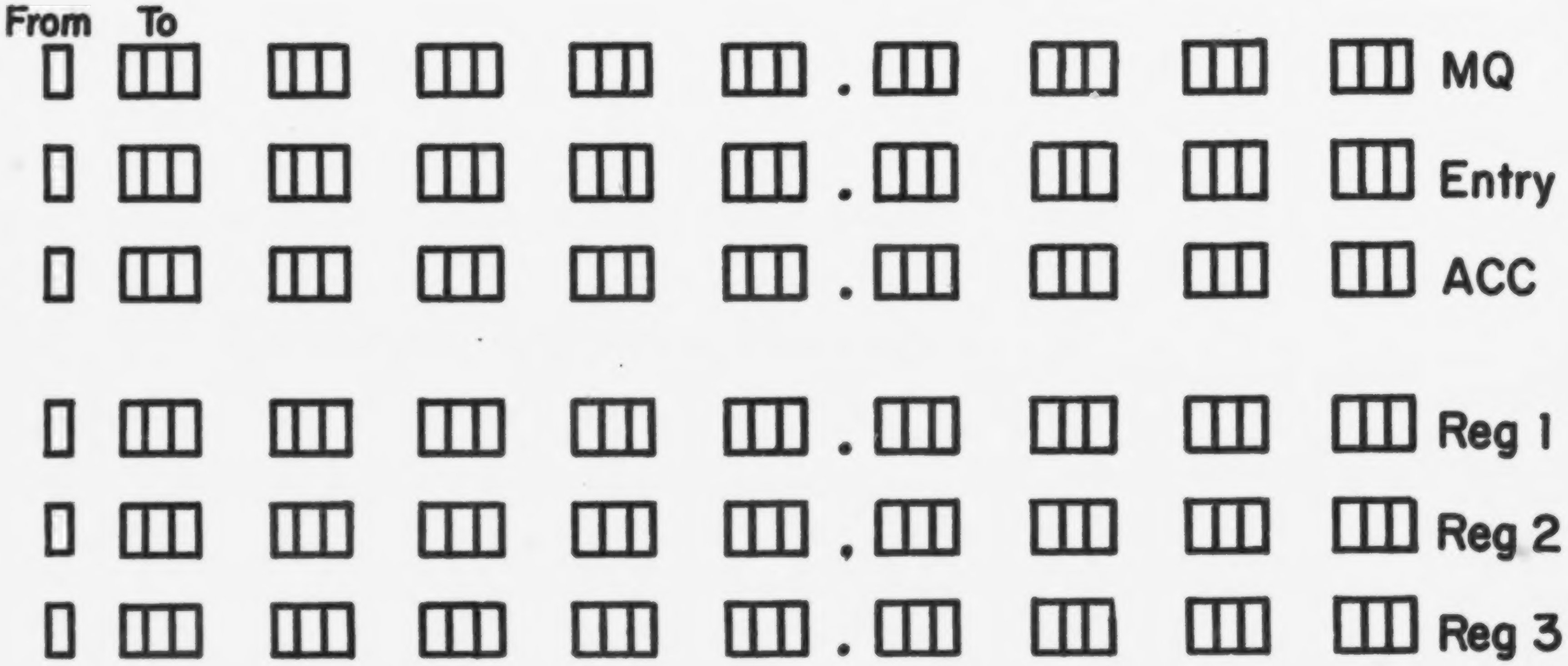
Subtraction: ACC − ENT → ACC

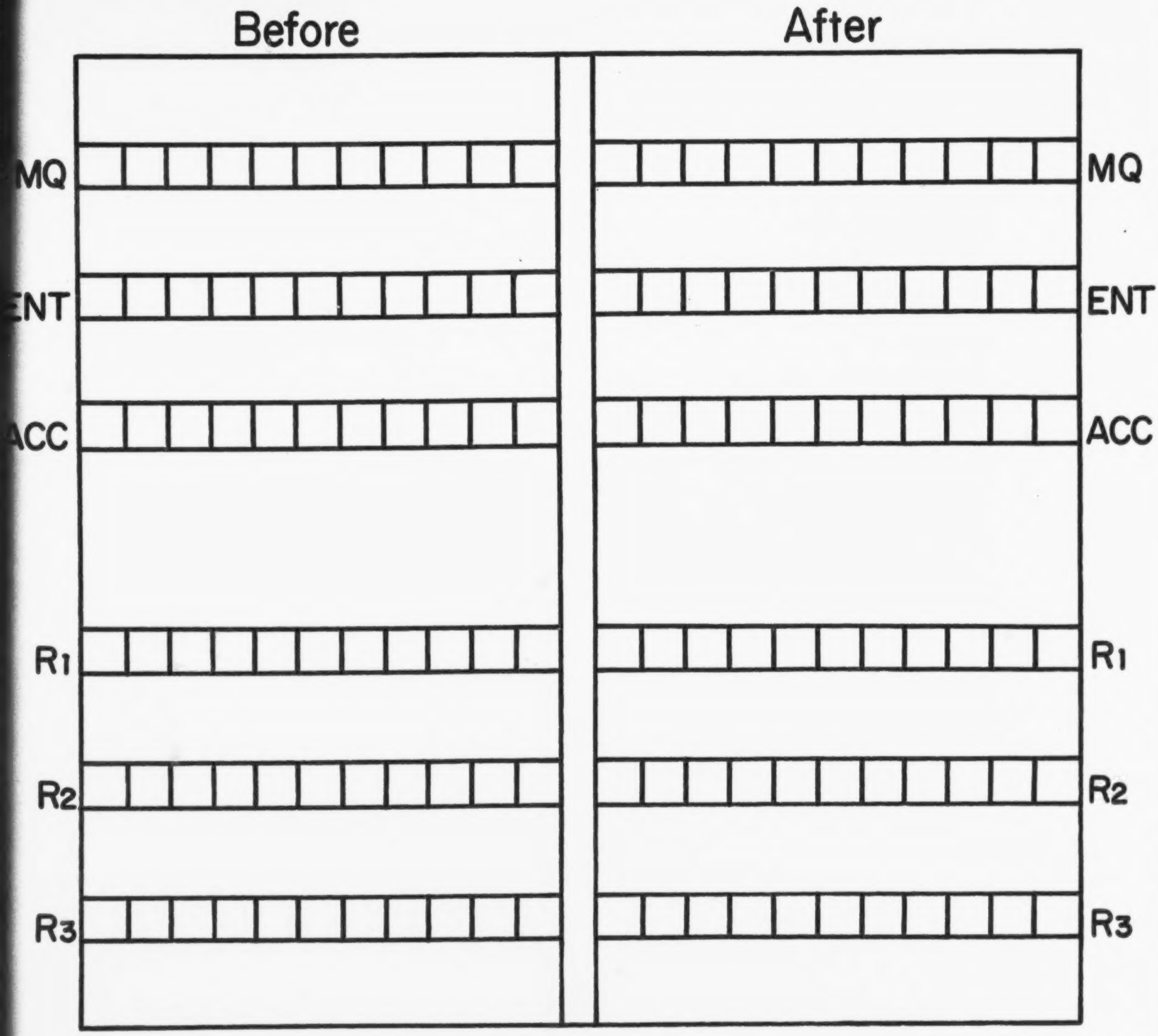
Multiplication: MQ × ENT → ACC

Division: ACC ÷ ENT → MQ


MQ
ENT
ACC
ARITHMETIC
R1
R2
R3
STORAGE


MQ
MQ
Entry
Entry
ACC
ACC
Reg 1
Reg 1
Reg 2
Reg 2
Reg 3
Reg 3
Transfer From ⟶ To
Fwd Space
Back Space
7
8
9
4
5
6
1
2
3
0
.
Shift ⟵
Shift ⟶
√
Sub −
Mult −
Mult +
÷
Add +
Clear & Mult
Clear Entry
Clear MQ
Clear Acc


From
To
MQ
Entry
ACC
Reg 1
Reg 2
Reg 3

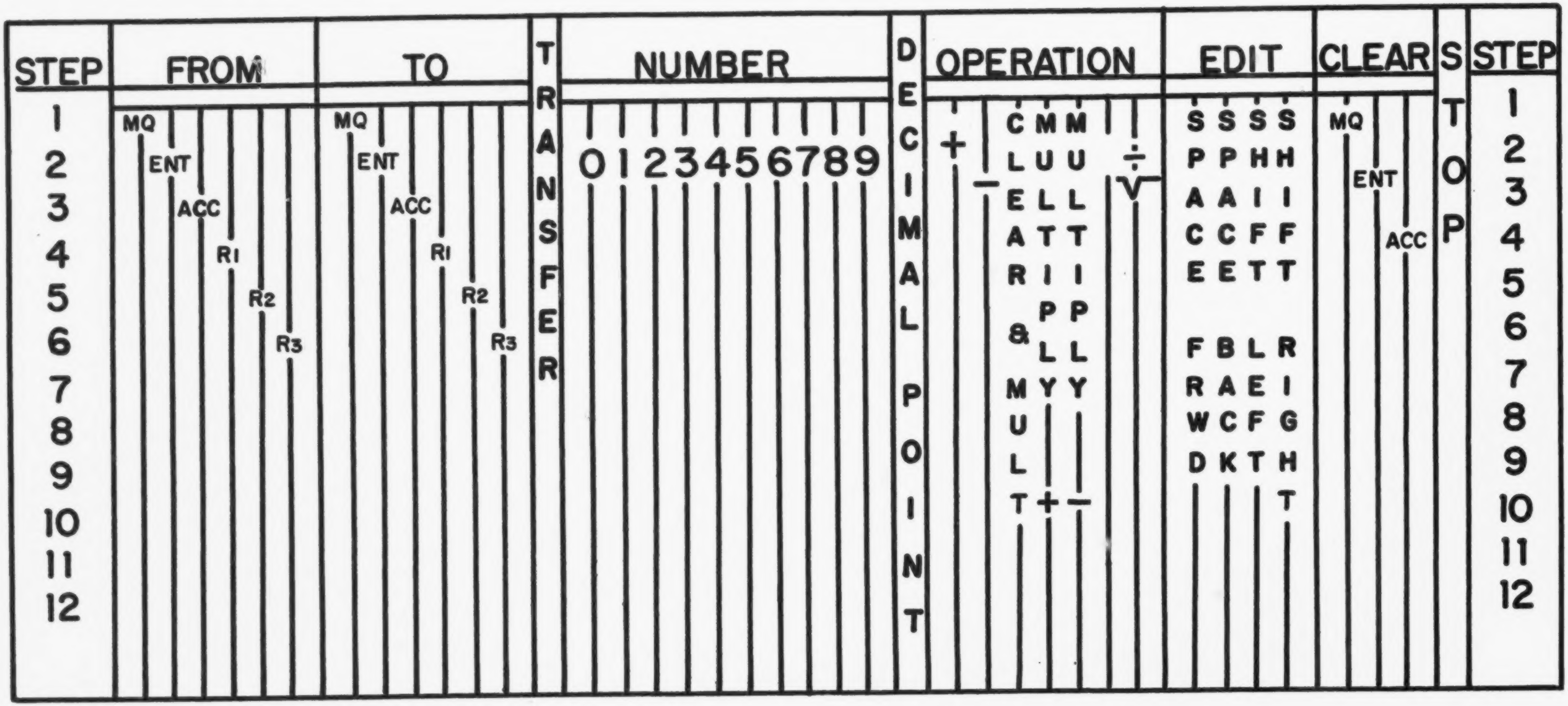

Before
After
MQ
ENT
ACC
R1
R2
R3

WYLE SCIENTIFIC PROGRAM CARD

STEP	FROM						TO						TRANSFER	NUMBER										DECIMAL POINT
	MQ	ENT	ACC	R1	R2	R3	MQ	ENT	ACC	R1	R2	R3		0	1	2	3	4	5	6	7	8	9	
1																								

OPERATION							EDIT				CLEAR			STOP	
+	−	CLEAR-MULT	MULT.+	MULT.−	÷	√	SPACE-FRWD	SPACE-BACK	SHIFT-LEFT	SHIFT-RIGHT	MQ	ENT	ACC		
															1

WYLE SCIENTIFIC PROGRAM CARD



YLE CODING SHEET
ROGRAM NAME ____________________

NAME ____________________

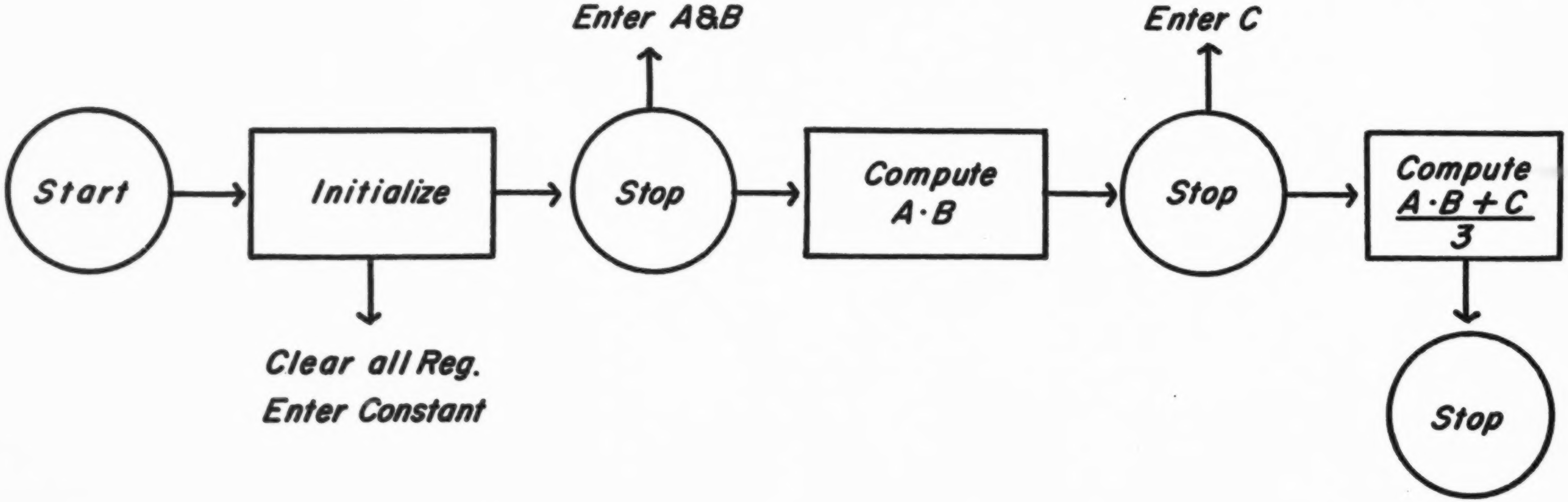
DATE ____________________

SHEET _____ OF _____

ARD#	ROW#	OPERATION	REMARKS
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	10		
	11		
	12		
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	10		
	11		
	12		

FLOW CHART

$$\frac{AB+C}{3}$$



YLE CODING SHEET
ROGRAM NAME $\frac{AB+C}{3}$

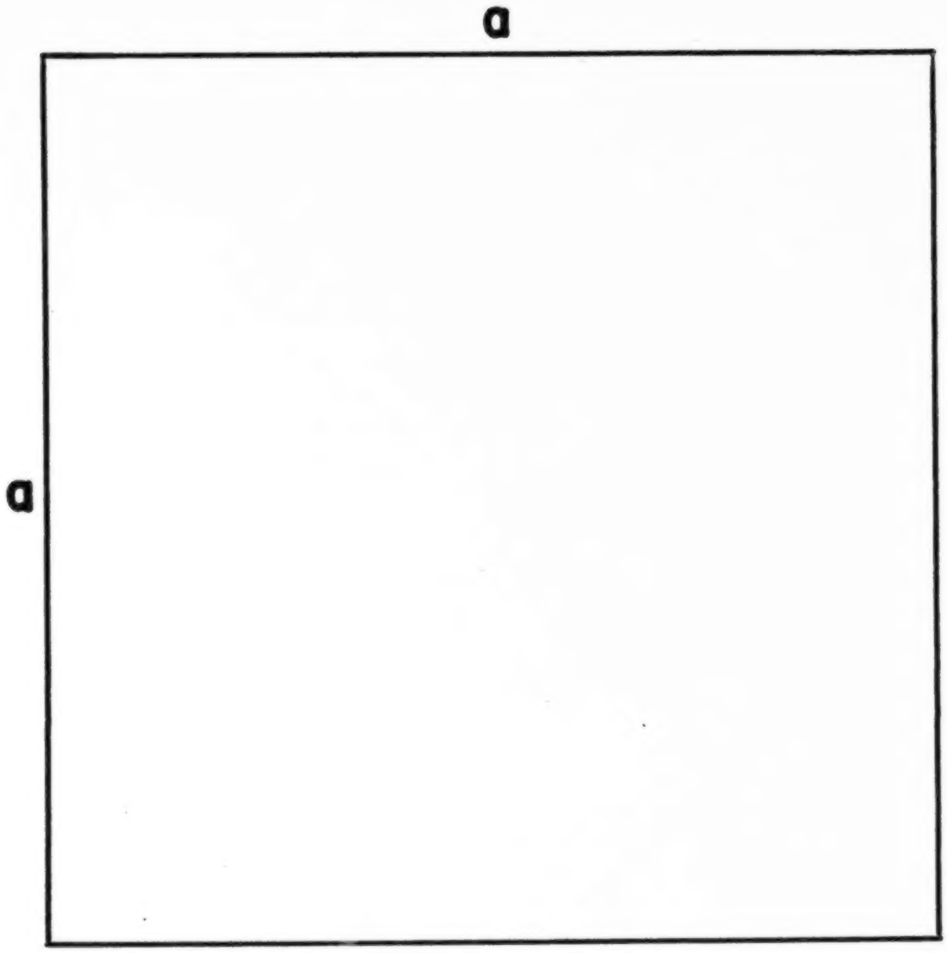
NAME John Doe
DATE April 8, 1967
SHEET 1 OF 1

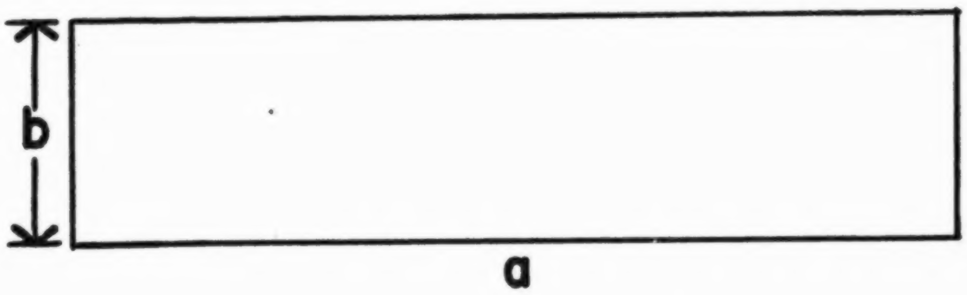
ARD#	ROW#	OPERATION	REMARKS
1	1	Clear MQ., ENT., ACC	
	2	To MQ	
	3	Stop	Enter A and dec. pt.
	4	To Entry	
	5	Stop	Enter B and dec. pt.
	6	Clear and Mul	ACC has $A \cdot B$
	7	To Entry	
	8	Stop	Enter C and dec. pt.
	9	Add	ACC has $A \cdot B+C$
	10	To Ent	
	11	3	
	12	Dec. Pt.	Entry has 3 and dec. pt.
2	1	Divide	MQ has $\frac{A \cdot B+C}{3}$
	2	Stop	
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	10		
	11		
	12		

PROBLEM

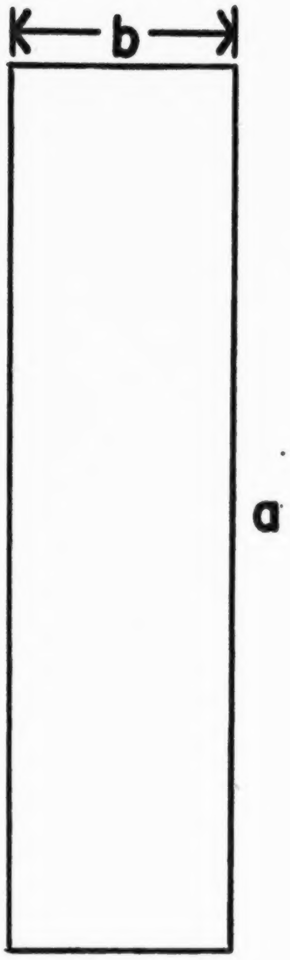
$$\left[\ \frac{AB-C}{3}\ \right]$$

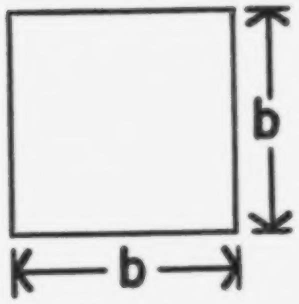
OPERATION	CONTENTS
clear	mq 0. ent 0. acc 0.
to mq	mq A ent 0. acc 0.
to ent	mq A ent B. acc 0.
to acc	mq A ent B acc C
mult +	mq A ent 0. acc AB+C
to ent	mq A ent 3. acc AB+C
divide ÷	mq (AB+C)/3. ent 0. acc 0.
stop	mq (AB+C)/3. ent 0. acc 0.

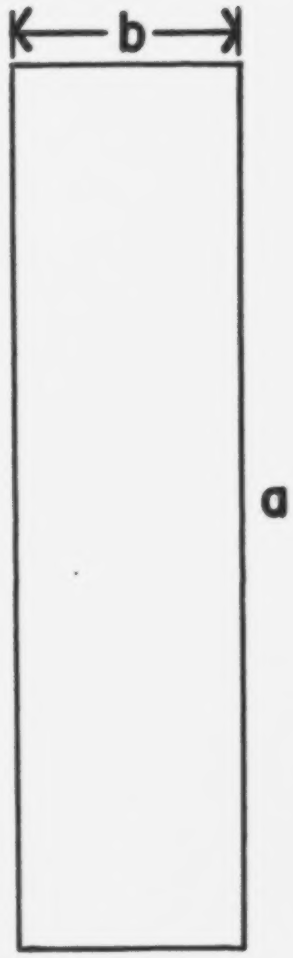

a
a



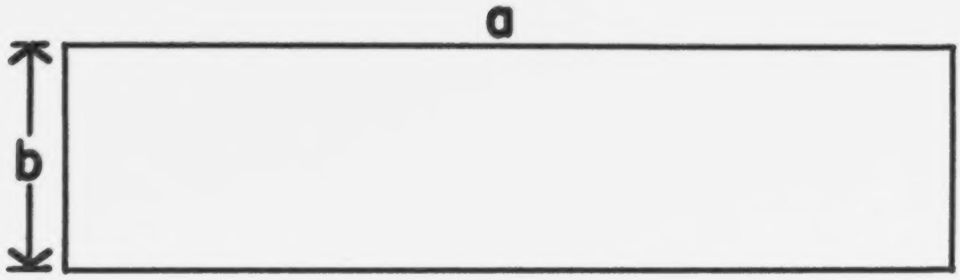
13b

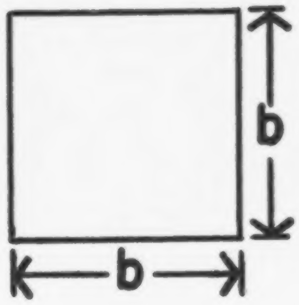

b
a

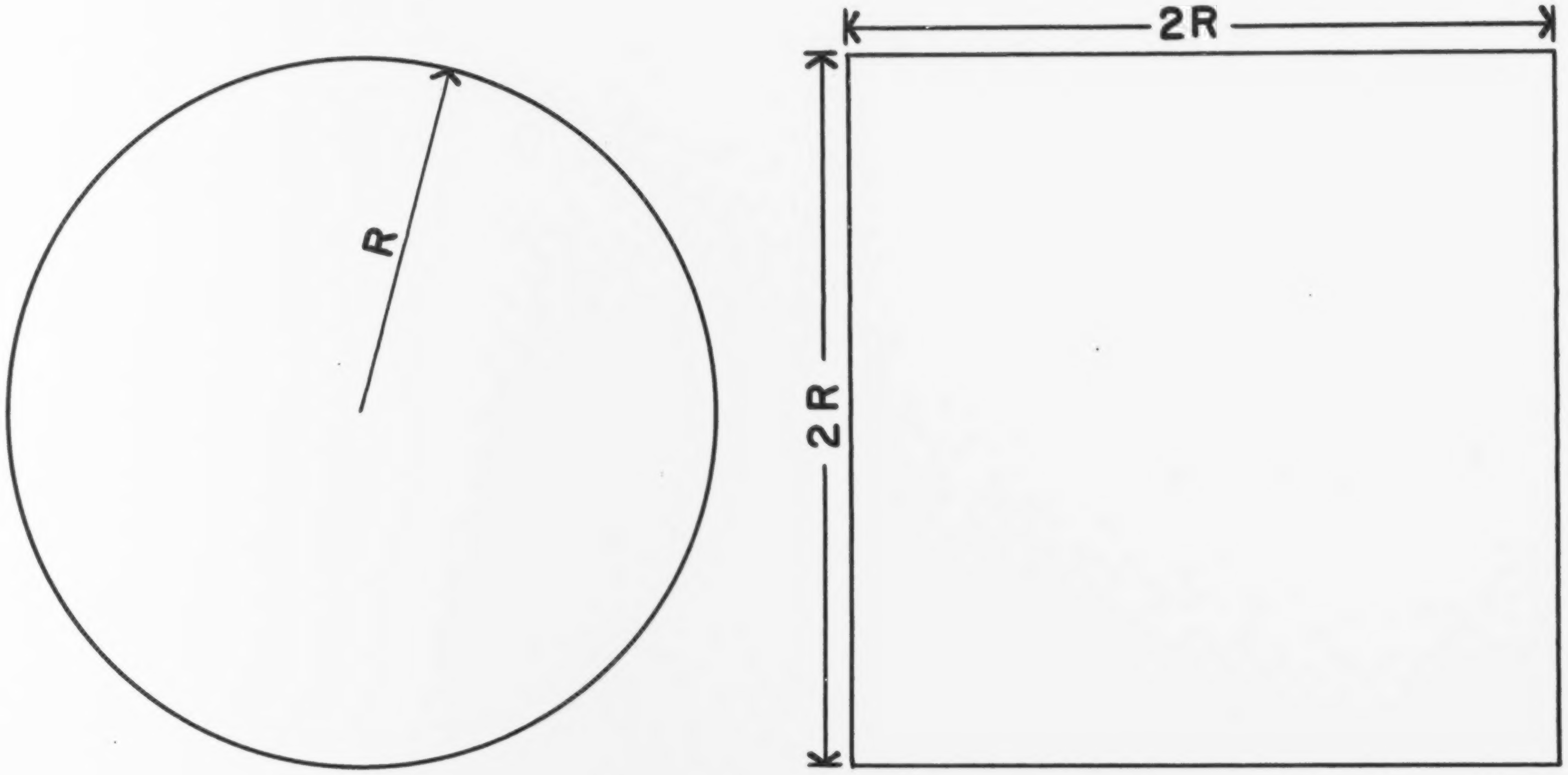

b
b



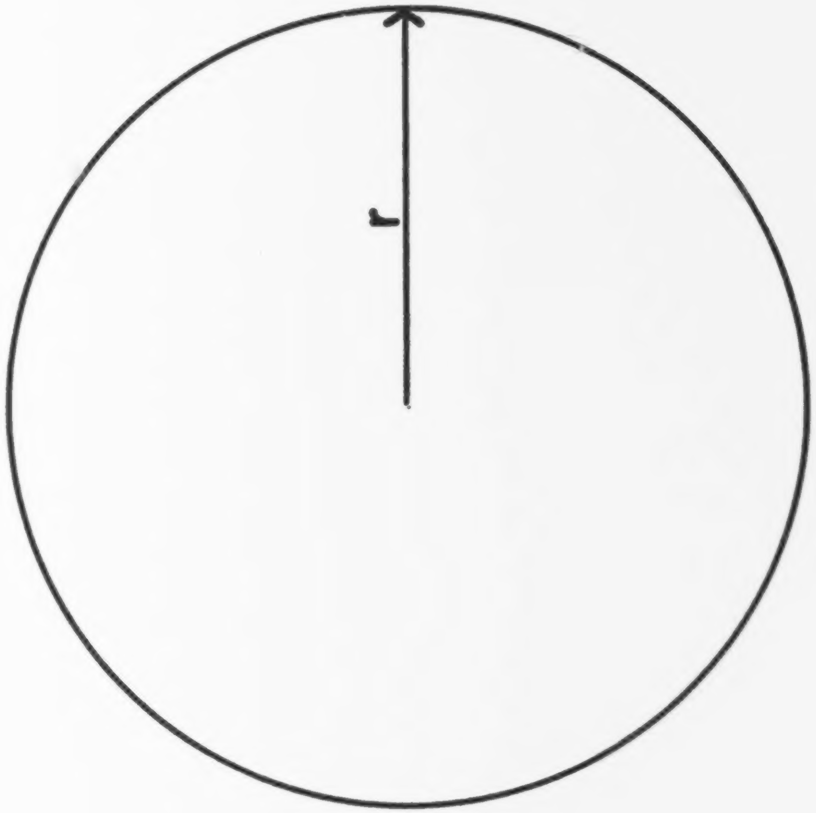
13e


a
b

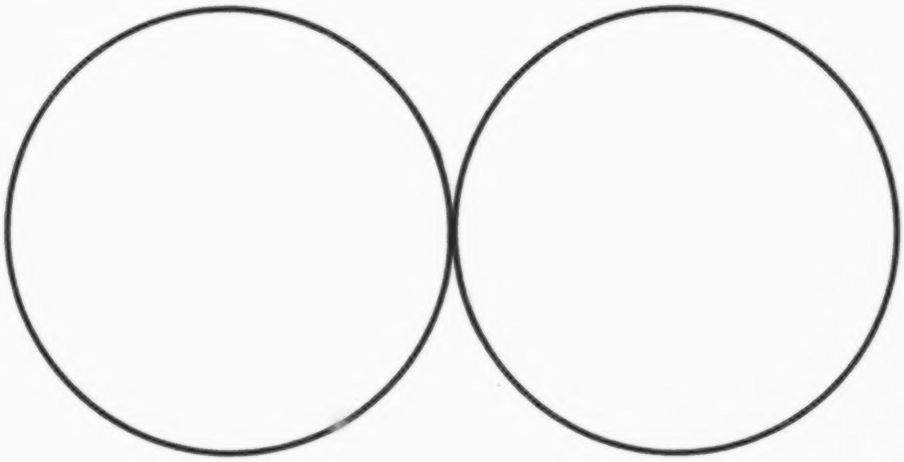

b
b



14a



14b



14c